A model of long-term learning: Integration of knowledge acquisition and knowledge compilation

Kurt VanLehn

Abstract

In order to understand how experience increases expertise, we propose to model the development of expertise in physics problem solving over long periods of training. The model will provide an explanation of 30 well-known phenomena, including expert-novice differences, practice effects and transfer effects. The resulting model should provide a unified theory of the acquisition of expertise in self-study settings.

Contents

1 Background

1.1 Knowledge acquisition and knowledge compilation

The current theory of cognitive skill acquisition distinguishes between knowledge acquisition and knowledge compilation. Knowledge acquisition amounts to understanding the subject matter, whereas knowledge compilation refers to the processes that turn the student's rough, shaky performances into smooth, confident ones. The usual assumption is that there are many knowledge acquisition process. Some common ones are reading a text for understanding, solving a problem by drawing an analogy to a previously solved problem or example, explaining an example to oneself, or asking a teacher for help when one is stuck. It is assumed that these processes are like cognitive skills in themselves, in that students learn how to do them at some point rather than being born with them. Moreover, there is no closed set of knowledge acquisition methods. The knowledge acquisition method depends on the form and content of the instruction, so new methods would arise whenever new forms of instruction are invented. In contrast, knowledge compilation mechanisms are usually assumed to be fixed parts of the cognitive architecture. Most theories propose only one type, or a few types of knowledge compilation mechanisms and try to demonstrate that they can explain all the relevant psychological phenomena.

This twofold view of cognitive skill acquisition contrasts with an older, more descriptive view that originates in studies of motor skill learning (Fitts & Posner, 1967). Skill acquisition is viewed as having three difficult-to-distinguish phases. In the first stage, the learners derive a basic understanding of the skill which is just barely adequate for solving simple problems. In the second stage, the learners gradually debug their understanding and extend it to solve harder problems. With sufficient practice, the learners enter the final stage, wherein they produce essentially the same solutions but with increasing speed and accuracy. According to the current theory, the first stage (understanding) is dominated by knowledge acquisition. The third stage (increasing speed and accuracy) is exclusively knowledge compilation. The second stage (debugging) is a mixture of knowledge acquisition and knowledge compilation.

It would seem on the face of it that the most interesting stage would be the second one, wherein knowledge acquisition and knowledge compilation interact to produce complex self-debugging behavior. However, no cognitive model has actually set out to model this stage. ACT* (Anderson, 1987; 1990), has never been implemented with both its chief knowledge acquisition method (analogical problem solving) and all its knowledge acquisition methods (proceduralization, compounding and strengthening), nor has it been applied to modeling second stage behavior. There have been some efforts in Soar (Newell, 1990) to model knowledge acquisition while reading (e.g., Pirolli & Bielaczyc, 1989; Lewis et al., 1989) and knowledge compilation while doing simple reaction-time tasks (Newell, 1990), but there have been no efforts that I know of to study what happens after an initial understanding of a task has been acquired.

There is nothing wrong with first developing models of acquisition and compilation separately, but they should eventually be integrated and tested. Historically, work on cognitive skill acquisition began with the expert-novice findings of the 70's. This led to models of knowledge compilation in the early 80's then models of knowledge acquisition in the late 80's and early 90's. At this point in time, we have a reasonably good understanding of the computational properties of knowledge compilation and knowledge acquisition as they function in isolation, so the scientifically appropriate next step is to understand how they would function when integrated. Taking that next step is exactly what the proposed research would do.

In addition, we need to re-examine the empirical claims made for knowledge compilation now that we have a better understanding of knowledge acquisition. The next section is devoted to a

preliminary re-examination that suggests that an integrated theory would substantially change our interpretations of many empirical phenomena that are currently believed to be caused by knowledge compilation.

1.2 Empirical evaluation of the three leading forms of knowledge compilation

Another important goal in evaluating the current theory is ascertaining its empirical adequacy. However, there is not just one theory, but many based on three rather distinct forms of knowledge compilation. Although a careful evaluation of any theory requires modeling the empirical findings in detail, the aim of the next several sections is a rough assessment of the state of the overall theory and of the empirical contrasts of the three specific kinds. This assessment will show that none of the three major models of knowledge compilation is sufficient to cover all the findings. The specific defects suggest revisions to the theory, which form the basis for the proposed research.

1.2.1 The 30 major findings on long term learning

However, before discussing the three models, the 30 major findings relevant to knowledge compilation are briefly presented. The three models of compilation will then be evaluated for their consistency with these 31 effects. Table 1 lists the 30 effects in the leftmost column. The other columns will be explained later.

There are 8 findings which are called basic effects because almost any layman could tell you about them. They are:

- Practice increases autonomy. With practice, the student relies less on external aids, such as textbooks or help sheets, and produces less behavior designed to facilitate their memory, such as writing an equation in its textbook form before applying it (Sweller & Cooper, 1985) or reciting a geometry axiom before applying it (Anderson, 1983).
- Practice increases speed. In fact, it often increases as the power of the number of trials of practice (Newell & Rosenbloom, 1981). However, this later finding is harder to explain, so it is broken out in Table 1 as a separate finding.
- Practice increases accuracy. It appears that both careless errors (slips) and errors due to misconceptions about the subject matter (bugs) decrease, although they may disappear at very different rates (Anderson, 1989).
- Practice decreases cognitive load, where cognitive load is measured by the ability to perform a dual task (Singley & Anderson, 1989; Shiffrin & Dumais, 1981). Experts can chat while they work and novices can't.
- Practice increases retention. The more one uses a skill or subskill, the harder it is to forget it and the easier it is to relearn it (Farr, 1987).
- Practice facilitates acquisition. It is easier to understand (acquire) a new skill when the subskills it uses have been mastered than when they have not yet received much practice (Bloom, 1984).
- The benefits of practice never cease. As far as anyone knows, speed, accuracy and the other benefits of practice never stop accruing even when one has practiced the same skill for years (Crossman, 1959).

Finding	Cases	Tuning	EBL
Basic effects			
Increases autonomy	-	-	+ +
Increase speed	+ +	+ +	+ +
Power law increase in speed	?	+	+
Increases accuracy	?	+ +	?
Decreases load	+	- -	+ +
Increases retention	+ +	-	+ +
Facilitates acquisition	+	-	+ +
Practice never stops helping	- -	- -	- -
Transfer			
Unit for predicting positive transfer	- -	+ +	+ +
Asymmetric transfer	+	+ +	+ +
Short term negative transfer	+	+	+
Long term lack of negative transfer	+ +	+	+
Practice increases specificity of transfer	+	+	+ +
Solution plans			
Basic approach task	+ +	-	-
Classification of problems	+ +	-	-
Mentioning problem or solution types	+ +	-	-
Strategy differences	+	-	+ +
Estimation of difficulty	+ +	-	-
Self-monitoring	+ +	+ +	-
Second-order features/classification	+	-	-
Abstract planning	-	+	-
Episodic memory			
Memory for problems	+ +	-	+ +
Memory for solutions	+ +	+	+ +
Semantic memory			
Similarity tasks	?	?	?
Order in free-recall	?	?	?
Semantic priming	?	?	?
Miscellaneous			
Generalization can be deliberate	?	-	-
Slow increase in new rule usage	+ +	+ +	?
Incomplete memory for puzzle solutions	?	-	?
Conceptual problem solving improves	+ +	+ +	+ +

Table 1: Consistency of 3 compilation models with 30 findings

These findings basically tell us that practice increases performance along all the dimensions that one would customarily use to measure expertise.

The main findings on transfer are listed below. All of these are documented fully in Singley and Anderson (1989).

- The amount of transfer between two tasks can be accurately predicted by the number of productions they share when knowledge for solving both is encoded as a production systems. That is, the unit of transfer is about the same size as a production.

- According to the usual measure of transfer, there is more transfer from a difficult task to a simple one than from a simple one to a difficult one. This apparent asymmetry is due mostly to measuring transfer by savings in learning time (i.e., $(T-t)/T$ where T is the time to learn a transfer task to criterion without prior training, and t is the time to learn it with prior training). If the same amount of knowledge is transferred in both cases, that knowledge is a larger percentage of the knowledge required to solve an easy task, so the savings in learning the easier task is greater. Asymmetry should perhaps not be listed as a separate finding, but as a correlate of the finding about units of transfer.

- In the short term, there can be negative transfer. For instance, if one has many years of experience in driving on the right side of the road, one tends to make mistakes for a while after starting to drive in England. Functional fixity and Einstellung are related effects. Carefully timed "be careful" hints suffice to prevent such negative transfer.

- In the long term, there is no negative transfer. The time to master a transfer task is still adequately predicted by production overlap even when the two tasks produce a great deal of short-term negative transfer.

- Practice increases the specificity of transfer. When subjects are given external material with generally useful principles, such as weight equals mass times gravity, and they are given varying amounts of practice on a particular application of the principle (e.g., given weight and gravity, find mass), then the transfer to a different application of the principle (e.g., given mass and gravity, find weight) is stronger when the subject has practiced less.

Most of these transfer effects can be modeled by assuming that small pieces of knowledge, about the size of productions, are the unit of transfer.

The next set of findings come primarily from expert-novice studies. They suggest that experts either have prestored plans for solving problems or can rapidly generate them. These studies have primarily been conducted with simple problems that both experts and novices could solve, so it is likely that the experts have seen these problems or similar ones before. Thus, they can be interpreted as the effects of having solved a problem before (i.e., having practiced that problem) on tasks that involve that problem or ones similar to it. Except for the last finding, which is rather recent, all these findings are discussed in VanLehn (1989).

- The expert can state a basic approach to solving a problem, whereas a novice gives platitudes or general advice (i.e., "I'd think carefully about the soughts." "I'd look through the book for relevant equations.").

- The expert can classify a problem according to the type of its solution, whereas novices can classify only on the basis of surface features.

- Both experts and novices tend to mention explicit problem categorizations early in the problem solving, but the novice mentions superficial categories (e.g., "Here's another inclined plane problem.") while the expert mentions solution-based categories (e.g., "Here's another conservation of energy problem.").

- In some task domains, but not all, there are distinct strategy differences between experts and novices.

- Experts are more accurate at estimating the difficulty of a problem than novices, where difficulty is measured by the subject's time to solve the problem.

- Experts give more self-monitoring statements than novices during problem solving, where a self-monitoring statement is a comment about the appropriateness of a method or the current amount of progress.

- Experts, and not novices, use second-order features when deciding how to classify a problem, where a second-order feature is something that is not immediately apparent in the problem statement but can be easily inferred. For instance, physicists note when a mechanics problem specifies both initial conditions and final conditions, as this suggests that the problem can be solved by conservation of energy.

- Koedinger and Anderson (1990) found that geometry experts do a form of abstract, visual planning before producing the proof itself. They say things like, "Since angles A and B are known and so is segment AB, I've got congruent triangles. That means I know everything about them. So I know angle C, which gives me angle D and hence another pair of congruent triangles, so I'm done." Koedinger and Anderson point out that because the experts planning knowledge deals with abstractions that are not present in the geometry axioms, most knowledge compilation methods cannot explain how it was acquired.

The bottom line is that most of these effects can be explained if experts have quick access to a solution plan. The plan allows them to state a basic approach to the problem, classify it, estimate its difficulty, and monitor their progress during the solution process. It is unclear whether experts retrieve the solution plans from memory or generate them on the spot.

Another set of findings, also reviewed in VanLehn (1989), concerns the fact that experts have better memory for episodic information that is relevant to problem solving. In particular,

- Experts can recall problems better than novices, but only if the stimuli make sense as problems. Pieces randomly strewn about a chess board are recalled equally well by both chessmasters and novices.

- Experts can recall solutions to problems better than novices.

Both short-term and long-term recall have been tested, as well as some long-term recognition.

Another set of findings, reviewed in VanLehn (1989), concerns the semantic memory of experts and novices. All the tasks use technical terms as their stimuli, as opposed to problems or solutions, so they therefore measure changes in the student's semantic knowledge.

- Experts and novices differ on their judgments of which words are most similar to other words.

- During free recall of lists of technical terms, experts group the items differently than novices.

- In the standard semantic priming paradigm, experts and novices differ on which words prime the recognition of which other words.

Basically, technical terms mean different things to an expert and a novice. The physics novice thinks that "weight" is more similar to "mass" than "force," whereas experts have the reverse opinion, presumably because they have assimilated the technical definition of weight. (Physicists define "weight" to be a force caused by gravitational attraction.) Not only do the denotations of terms differ, but subtle connotations differ as well.

Lastly, Table 1 shows four miscellaneous findings. These are included for completeness but have no great bearing on the argument and thus will not be discussed further.

1.2.2 Case-based and schema theories: Problem-sized units of transfer

It is important to carefully compare the various models of knowledge compilation to the empirical findings, because there is a common perception in the field that knowledge compilation is a solved problem in cognitive science, and has been solved since the early 1980's when ACT*, Soar and other knowledge compilation systems made their first appearances. However, most of the arguments of consistency with empirical finding from that early era were made in a general way with no reference to specific data and no simulation runs. This leaves some uncertainty as to whether even the espoused explanations would hold up. Moreover, the empirical work did not stop ten years ago, although it did die down a bit, and there are some new findings to account for. Lastly, much has been learned about knowledge acquisition since the early 80's, and it now appears that phenomena that were once thought to be caused by compilation may in fact be due to acquisition. Thus, a close examination of the empirical adequacy of standard theories of compilation is timely and indeed perhaps even necessary for further development of our understanding of cognitive skill acquisition.

Unfortunately, this proposal is too brief to allow even a careful discursive examination of the models. Part of the work to be done will be drafting a review article arguing in more detail the case to be glossed here.

There are three major kinds of knowledge compilation. This section discusses the first one, which is based on a simple idea: When you have finished solving a problem, store in memory a description of the problem and a description of your solution, perhaps generalizing each somewhat. Just before solving a problem, consult your memory to find a similar problem, and if one is found, solve the present problem by "replaying" the old problem's solution. This is the essential idea behind two well-known kinds of knowledge compilation, case-based reasoning and schema theory. These two idea differ mostly in the level of generality of the problem-solution pairs that are stored. Case-based models store the specific problem and its solution, with little or no generalization. This means that finding and using a case requires a kind of analogical transfer. On the other hand, according to schema theory, the experience of solving a problem is generalized before storage in memory. To use a schema, one matches its problem template to the current problem, thus creating a mapping between the objects in the problem and the slots in the template, then applies that mapping to its solution template, thus creating a solution plan that is specific to the given problem. Computationally, the match-and-apply process is almost identical to analogical transfer, so the only real difference is in the level of generality of the stored problem-solution pairs. Since that different is not important to the review being conducted here, case-based and schema theories can be treated as the same thing. The key idea of both these approach is that the unit of transfer is large: a whole problem and its solution.

The second column of Table 1 shows how well the problem-sized unit of transfer approach fares. A "++" means that the theory explains the finding. A "+" means that the theory alone will not explain the finding, but assumptions can be added that allow the theory's processes to explain the finding. A "?" means that the predictions of the theory are unclear. With some reasonable assumptions, the theory might explain the findings, but with other equally reasonable assumptions,

it might not. A "−−" means that the theory predicts the exact opposite of the finding, and that there is no way to avoid the misprediction without gutting the theory and making it into one of the other theories. A "−" means that the theory offers no explanation of the findings, but that ancillary mechanisms can be added that will allow an explanation, and that theory's mechanisms neither help nor hurt the ancillary mechanism explanation.

The forte of this theory, which is based on importing whole solutions from the past to the present, is that it offers ready explanations for the findings listed under "Solution plans" in Table 1. The essential assertion is that practice causes accretion of a large library of cases/schemas, and that experts but not novices can therefore retrieve a case/schema for most problems. This allows them to state a basic approach to solving the problem, classify it on the basis of its solution, estimate its difficulty and monitor progress during the implementation of the solution plan. Strangely, the theory seems to predict that the novice and expert would use the same strategy, since the expert is merely reusing a solution generated in the past, and thus by induction, when the expert was a novice. However, Elio and Scharf (1990) have worked out a way to represent solutions as sets of inferences rather than sequences, and this allowed them to demonstrate strategy shifts as well as the emergence of second-order features during classification. This means that the only solution plan finding that is truly outside the competence of the theory is the Koedinger and Anderson finding that experts in some domains do not retrieve solution plans, but built them on the spot.

On most of the other findings, the explanatory competence of the theory is spotty at best. It is particularly bad at explaining transfer effects. The basic problem is that case-based and schema-based theories predict that transfer only works well when the two tasks have very similar solutions. The fault lies in the computational problems of adapting an old solution for a new problem. It can involve disassembling the old solution, discarding some but not all of it, devising replacements for the discarded parts, and reassembling it. The complexity of this process comes out most clearly in applications efforts, such as those of Kambhampati (1990), where the investigator does not have the luxury of choosing the cases that best demonstrate his or her adaption mechanisms. Because the replaying process is so complex and fragile, this theory predicts a much sharper drop off in transfer than the ones actually observed. Whereas Singley and Anderson (1989) repeatedly find that the degree of transfer is linearly related to the overlap in productions, the large-unit transfer theory would predict a non-linear relationship: A few differences between the task would allow nearly complete transfer, so the second task can be accomplished with virtually no learning. However, if there are more than just a few difference, transfer becomes impossible and the second task has to be learned from scratch. To put it intuitively, the large-unit theory predicts that experts will excel intermediate (who have the domain knowledge but haven't practiced it much) only when solving problems that are very similar to ones the expert has seen before. If both experts and intermediates are given a truly novel problem, they will both flounder. This just doesn't seem like expertise. So we must conclude that schema and case-based theories are, in themselves, not adequate to explain all the findings in the literature.

1.2.3 Search tuning

The second major approach to speeding up problem solving is to invent and save search heuristics so that bad paths in the search space can be avoided in the future. This technique is often called search tuning because the learner "tunes" or improves the search without changing its essential character. The earliest program of this type is Samuel's checker playing program (Samuel, 1963), which learned to play an excellent game by figuring out heuristics for evaluating proposed board positions. All other programs have learned heuristics that choose operators or goals. For instance, Lex (Mitchell et al., 1983), and Sage (Langley, 1983) examine their search trees and find out where

they diverged from the solution path. Let's say that the right move at that point was to apply rule 1 and the wrong move was applying rule 2. They figure out what the problem state was at the time of the bad choice, and build a heuristic that says, in essence, if the problem state is similar to this one, then choose rule 1 and not rule 2. Lex and Sage build general heuristics, but this is not the only way to tune a search. Analogical search control (VanLehn & Jones, in press-a; Jones, in press) merely stores the good rule choice associated with the state in which it was made. Analogical retrieval and matching are used to access these saved pairs and use them to make decisions about which rule to apply. Prodigy (Minton et al., 1989) builds heuristics by proving that a given choice is good or bad, then convert the proof to a search heuristic using EBL.

As Table 1 shows, search tuning can explain a few of the basic findings, namely the increase in speed and accuracy, because it learns to avoid bad paths in the search space. It also does a good job explaining the transfer findings because it's unit of transfer is a small one: the search heuristic. Thus, transfer is roughly proportional to the overlap in the search spaces of the two tasks.

However, search tuning has little to contribute to explaining the other findings. Although experts can access solution plans, search tuning has no mechanism that would generate such plans. Search tuning has nothing to say about the memory findings, nor about the more mnemonic of the basic effects. The best it can contribute is a rather far-fetch explanation of the expert's better memory for problem solutions: The experts' optimized search allows them to regenerate the solution plan quickly enough that it passes for retrieval when in fact is is an extreme form of reconstruction.

1.2.4 Explanation-based learning

Explanation based learning (EBL) is the third form of knowledge compilation. The basic idea is to analyze a sequence of inferences and create a new inference rule that will achieve the same effect in one step. If the problem solving is goal-driven, then EBL is typically used to compress the sequence of inferences or operator applications that achieved a goal into a single rule or macro-operator that achieves the goal in one step.

There are many EBL systems. The most psychologically oriented ones are Soar (Newell, 1990) and ACT* (Anderson, 1983, 1990). When Soar accomplishes a goal, it notes the information that achieved the goal and the state that was present when the goal was set. It traces backwards through its records of rule firings to figure out which aspects of the state are responsible for achieving the goal. These aspects of the initial state become the condition side of a new production rule. The information that achieved the goal becomes the action side of the new rule. In this fashion, Soar builds a new production rule that does the same work that multiple production rules did. ACT*'s production composition is similar. It also applies when a goal is accomplished, and it also forms a single production rule that accomplishes the work done by multiple pre-existing production rules.

EBL is a little like schema theory, in that both form a pair. One member of the pair is matched to the current problem (schemas) or goal/state (EBL), and if the match succeeds, the other member of the pair, the solution plan (schema) or macro-operator (EBL), is applied in the expectation that it will achieve the goal or solve the problem. One difference between EBL and schema theory is that schemas are formed only for top level goals which solve whole problems, whereas EBL can be applied to any goal. This is not a necessary feature of the two techniques, but we will pretend that it is so that the distinctions between the two remain as sharp as possible. Another difference between EBL and schema theory is that schemas have multiple-step solution plans whereas the equivalent object in EBL is supposed to execute in a single step. This later difference is fundamental, and makes it inherently more complex to explain practice effects with EBL systems. Consider someone who has just learned, by trial and error, the right way to set the time on his new TV. A schema system would store this as a plan, whereas an EBL system would store it as a new rule. If one

later wanted to set the time, one would retrieve the schema and execute the plan by pushing the right buttons. An EBL system would retrieve the rule and execute it in a single step. This just doesn't make sense, for one can't execute a sequence of button presses in a single step. Although Anderson never noticed that ACT* shouldn't compound productions that contain motor actions, Newell noticed that Soar shouldn't chunk over physical actions, so he proposed a solution. A properly programmed Soar system constantly asks itself at the top level, "What action should I take now?" Because it usually doesn't know, it must think before deciding. With practice, these deliberations before each action get shorter and shorter, so Soar speeds up.

The EBL theory of knowledge compilation excels at explaining basic effects. It is unique among the three models in that it predicts a decrease in cognitive load due to practice. This occurs because EBL tends to flatten goal hierarchies. A single rule application replaces a whole goal-subgoal tree of rule applications. Thus, the average number of goals held in working memory decreases, which explains the ability of experts to use their surplus capacity to perform a dual task. EBL also provides a nice account for the increase in autonomy with practice. Just as Soar and ACT* would naturally tend to chunk over motor actions, they tend to chunk over perceptual actions. But this is good. The next time they want that piece of information, they do not have to perceive it in the external world, they can get it by firing a chunk what was formed by chunking over the perceptual action. Thus, experts don't have to "look at the book" as much as novices.

The EBL theory provides an explanation for most transfer effects. Because the units of transfer are production rules (or chunks), the theory explains the linear relationship between transfer and the overlap in task production sets. This allows the theory to explain also the findings of asymmetric transfer. To explain the appearance and disappearance of negative transfer requires assuming that chunks are initially too general (thus yielding negative transfer in the short term) and later are made more specific (thus blocking negative transfer). ACT* already has mechanisms (production activation and strength) that will achieve explain these effects, and Soar could probably be augmented to explain them as well.

The EBL theory is basically useless for explaining why the experts have access to solution plans and novices do not. Given Newell's trick for preventing chunking over motor actions, practice tends to make an EBL learner into a reactive system. In the limit, it becomes a stimulus-response machine: At each moment, a single chunk (S-R bond) matches the perceptible situation and advises an action, which the agent takes, thus changing the situation and leading to another SR cycle. This is not bad in itself, since an expert chess player is usually also very good at lightening chess, and expertise in some skills, such as flying aircraft, is essentially reactive. However, experts even in these skills seem to have plans which in more leisurely circumstances they can generate and use. Although it would run against the grain of EBL systems, I believe that a new mechanism for learning plans could be added to them in order to explain the findings about solution plans. Hence, these findings receive only a "–" in Table 1 rather than the "––" that their natural inclinations would warrant them.

1.2.5 Summary: There are really only two puzzles

The discussion showed that case-based and schema-based models have trouble explaining transfer effects, that EBL-based models have trouble explaining solution plan effects, and that search tuners have trouble with explaining all findings other than a few basic effects and some transfer effects.

In addition, there are some effects that none of the models explain. One of the most telling is that none explain how people can continue to improve in speed and accuracy even after years of practice. Even if goal hierarchies and search are assumed to exist at the motoric and perceptual levels, these learning mechanisms would sooner or later build all possible chunks, learn all possible

cases and form all possible search heuristics. It seems that only a continuous quantity, such as trace strength or its neural equivalent, could explain why learning never stops.

Human memory is strange thing, and semantic priming is one of its strangest properties. None of the model of practice can account for it directly, nor any of the other related findings about the semantic relatedness of technical terms. One might assume that when a chunk (or schema, or search heuristic) mentions two terms, then activation can flow between them. Thus, with more practice, more chunks mention the two terms, and the more semantically connected they become as measured by semantic priming and the other tasks. It is not at all clear whether this would work, so these findings receive a "?" rating in Table 1.

All this suggests that there are really only two puzzles in the knowledge compilation field. The first is to explain how experts can access solution plans, even when those plans are not simply related to novice problem solving (Koedinger & Anderson, 1990). Case-based and schema-based models of knowledge compilation provide a partial solution to the puzzle, but they are not sufficient alone to cover all the other empirical findings.

The second puzzle is to explain what one could loosely call memory effects. These include the findings listed under the episodic and semantic memory categories in Table 1 as well as the increase in autonomy with practice (along with the strongly related finding about the increase in specificity of transfer with practice), the decrease in cognitive load, and the increase in retention. These are all classic memory phenomena that happen to have been studied in cognitive skill acquisition as well as their home territory of verbal learning. There are probably other robust verbal learning phenomena that could be detected in cognitive skill acquisition as well. One scientifically appropriate strategy is to try to show that a single model of memory can account for both traditional learning phenomena and the memory effects of cognitive skill learning. Newell (1990) and Anderson (1983) have done just that.

However, another scientifically appropriate strategy is to find a way to model cognitive skill acquisition while making only minimal assumptions about memory. Just as one can have successful chemical theories that make only minimal assumptions about quantum mechanics (e.g., the Bohr atom), it would be scientifically productive to search for a "molar" theory of cognitive skill acquisition and let other investigators (probably connectionists) develop an "atomic" theory of memory. This is the approach to be followed in the proposed research. Thus, the memory puzzle is to find the appropriate molar assumptions to make about memory that will allow a theory of cognitive skill acquisition to explain its findings while leaving the memory effects to be restated (one can hardly say "explain") by the molar assumptions.

2 Research Objectives

The preceding section argued that there are essentially two problems with the contemporary theory of cognitive skill acquisition. First, no one has seriously tried to simulated long term learning that includes both knowledge acquisition and knowledge compilation. Second, current theories of knowledge compilation can account for either planning effects or some memory effects, but no theory can account for all the long term learning effects.

There are two objectives of the proposed research. The first is to remedy the incompleteness documented above by building an integrated model of knowledge acquisition and knowledge compilation, and demonstrating that it can account for most if not all of the long-term learning effects.

The second objective is more subtle and more difficult to introduce because the empirical work has not yet been done in a systematic fashion. Basically, after several years of analyzing protocols of subjects learning physics and other skills, I have yet to see any very profound behavioral change

that did not actively involve the subject constructing new knowledge. For instance, in reanalyzing the Tower of Hanoi protocols of Anzai and Simon (1979), I found eleven distinct events where the subject apparently learned something new about the puzzle (VanLehn, 1989). In most of the events, the subject's comments indicated awareness of the learning and suggested that a methodical plan for discovering the puzzle's secrets was being followed, complete with experimentation. All this discovery learning is more like knowledge acquisition than knowledge compilation. Yet, in this protocol, the subject solved the same puzzle multiple times without feedback, so her learning should have been the product of knowledge compilation instead. Others have studied learning in similar situations where lack of instruction should defeat knowledge acquisition and allow knowledge compilation to shine through. Siegler and Jenkins (1989) studied children learning simple addition strategies. Kuhn et al. (1988) studied college students learning experimental design skills. In both cases, the investigators found distinct learning events, which they interpreted as the result of some form of conscious knowledge acquisition. Of course, not all the subjects commented about all their discoveries. Sometimes the only accompaniment of a learning event was a long pause. Subjects' retrospective descriptions of what they learned and how they learned it were often incoherent. Nonetheless, it appears that their learning was more likely a form of knowledge acquisition than knowledge compilation.

More recently, I have been analyzing protocols of subjects learning physics by studying examples and solving problems (VanLehn, Jones & Chi, in press). Although the emphasis in these analyses was on finding knowledge acquisition events, I could not help noticing that there was only one obvious signs of knowledge compilation. It was the autonomy finding of Table 1. Subjects gradually ceased to refer physically to the examples, although they still mentioned them and were apparently referring to their memories of the examples instead. They also slowly ceased to write "$F = ma$" or "$w = mg$" on their worksheets before substituting formulas appropriate to the problem for the variables in these equations (cf. Sweller & Cooper, 1985).

In preparation for this proposal, I have been analyzing protocols collected in 1982 by M.T.H. Chi. Two subjects attended physics classes then solved physics problems during 16 sessions of 4 problems each. Problems were often repeated in different sessions, so there was ample opportunity to observed changes in problem solving caused by practice over the approximately 20 hours of training. The study was supposed to be like the Anzai and Simon one, in that the experimenter was not supposed to help the subject, give feedback or otherwise disrupt the classic self-studying situation wherein pure practice effects should surface. Unfortunately, the experimenter interjected feedback and tutorial help rather frequently, albeit sometimes unintentionally. Hence, these data were transcribed but not analyzed further.

I decided to examine these protocols and try to factor out the knowledge acquisition caused by the experimenter. Given the research since 1982, we now know roughly what an acquisition event looks like in protocol data, so this factoring out seems feasible. Although my analyses have been too unsystematic to present here, the bottom line is that when knowledge acquisition is factored out, there are few changes in behavior left for knowledge compilation to explain. One is the increase in autonomy, which was noted in the other physics protocols. The other is more difficult to describe. Roughly speaking, early in their training, the subjects seem to waffle for several seconds before taking most steps. Later in their training, this waffling disappear, and they moved quickly from one step to the next. This is not to say that they knew what to do. The problems gradually increased in difficulty, and the subjects were just as likely to say that they were not sure they were on the right track when they were doing easy problems at the beginning of the study as when they were doing hard problems at the end of the study. At this point, I don't know what causes the waffling and I don't know what makes it gradually disappear. However, I am fairly certain that it's disappearance is not caused by learning new physics rules. I think is it the result of some kind of

knowledge compilation.

These analyses suggest that the traditional knowledge compilation mechanisms are much too ornate. Most of the effects that they are supposed to generate just don't show up in protocols except when accompanied by signs of knowledge acquisition or plausible triggers for knowledge acquisition (e.g., feedback from the experimenter). This leads to the conclusion that many of the long term learning effects may really be caused by deliberate, conscious knowledge acquisition events. True knowledge compilation may be limited to relatively simple memory effects, such as the increase in autonomy with practice.

Finally we come to the second objective of the proposed research, which would only be tackled if the funding period were extended from the proposed one year to a total of three years. The second objective is to generate systematic protocol data on long term learning and model it with an integrated learning system. This should help us understand in much more detail how knowledge acquisition and compilation interact and which one is responsible for which sorts of behavioral changes. It also should help us create a theory of long term learning that is more grounded in the day-to-day reality of learning. For instance, it should explain the painfully apparent phenomena of waffling and its gradual dissipation. To put it bluntly, who cares if a model of knowledge compilation can explain the power law to three decimal places when it can't even explain phenomena that confront teachers, trainers and tutors every day.

In order to make this research feasible, two major restrictions in the scope of work will be applied. First, we will only work with one task domain: physics. This task domain is fairly well understood because many of the empirical findings have been produced with physics as the task domain. We also have a running program, Cascade, that models knowledge acquisition in physics (VanLehn, Jones & Chi, in press). We also have a running system, Olae, for collecting protocol and other data on subjects' physics knowledge, and it is being extended to automate some parts of the data analysis (VanLehn et al., in prep.). Lastly, physics is an extremely rich task domain in that it has some math, some difficult technical concepts (e.g., force), a rich interaction with common sense and some mental modeling. It is also typical of many other scientific fields, such as thermodynamics, aeronautics or electronics, where the introductory courses concentrate on teaching students how to mathematically analyze physical systems. Thus, the results from studying physics should generalize straight forwardly to some other task domains, and perhaps even further.

Another restriction is that we will work only with subjects learning in a single instructional situation: self-studying of printed materials. Knowledge acquisition methods vary considerably depending on the kind of instruction available to the subjects, so sticking with one type of instruction is a practical necessity for a project that aims to model both knowledge acquisition and knowledge compilation. Self-studying is a good choice for this work because it is also well-studied (e.g., Anderson, Farrell & Saurers, 1984; Chi, Bassok, Lewis, Reimann & Glaser, 1989). Unfortunately, my recent protocol analyses suggest that it is extremely slow and frustrating for subjects to learn when they have no feedback at all, so we may have to introduce some kind of simple feedback, such as allowing them to look at the answers in the back of the book after producing their first answer. The 1982 protocols showed that subjects adapted readily to tutorial feedback and even began to depend on it. This suggests that too much feedback can cover up interesting learning effects, so the proposed studies will not use the immediate feedback regime used in Anderson's studies.

2.1 Implications for machine learning

AI has built some impressive expert systems, and machine learning has generated a whole technology for acquiring and improving knowledge. One would think that someone would have built a machine learning program that learns to be an expert system. But no such system has been built.

This fact has been ruefully pointed out by several keynote speakers at the last few machine learning conferences (e. g. Dietterich at the 1989 workshop), but so far no one has risen to the challenge.

The proposed research will build a system that achieves expertise in a very narrow slice of physics. It should be as competent as Mecho, one of the first expert systems in AI (Bundy, et al., 1979). Thus, it is a step in the right direction to meeting the knowledge acquisition challenge.

However, it is perhaps more important to machine learning in advancing a line of research that Tom Dietterich, in his invited address to the 1989 machine learning conference, distinguished as one of the most promising unexplored areas in machine learning. Dietterich pointed out that most current machine learning approaches to acquiring knowledge for expert systems try to do it the hard way, by inducing the knowledge from data. This is not how most experts learn. They learn from instructional material and from other experts. Only in the last stages of their training, if at all, do they outrun the prepared sources of expertise and begin to draw their expertise directly from interaction with the problem solving environment. Most machine learning approaches to knowledge acquisition take the discovery learning approach from the very start, so of course they do not get very far. Dietterich claims that machine learning would go much further in acquiring expertise if it simulated the learning of human experts-in-training. That is exactly what the proposed research will do.

Lastly, this work would continue a line of investigation into felicity conditions as expediters of machine learning. A felicity condition is a design constraint on instructional material. Material meeting that constraint will be easier for certain learning agents to learn from (VanLehn, 1987; 1990; Hall, 1988; Winston, 1975). Theoretical work (e.g., Rivest & Sloan, 1988) suggests that the only way to learn rich knowledge from example-laden instruction is to design a combination of a learner and the appropriate felicity conditions for it, then write instruction that obeys the felicity condition. This has not been a very actively pursued approach to acquiring expertise, perhaps because researchers think that the effort of writing instruction that obeys felicity conditions is tantamount to programming the knowledge directly into the agent, rather than having the agent learn it itself. However, if the instruction is already written, then this is not a problem. It appears that there may be some felicity conditions that could benefit an agent learning technical material like physics (VanLehn & Jones, in press). It is an open question whether all technical material obey these felicity conditions, or even whether physics instruction does.

2.2 Towards pedagogical applications involving simulated students

In addition to the scientific benefits, which have been stressed above, there are some potential pedagogical applications of the proposed research. This research will move the field a step closer to being able to simulate students with enough fidelity that they can be used in place of real students. There are a number of applications where such a technology would be useful.

Most obviously, simulated students can be used to train teachers. Simulation-based training has a good track record, especially for training military and industrial skills. In general, simulators make it easier for the student to learn because they allow the student to halt the simulation, replay parts of it, reflect on their performance post-facto, run time faster or slower than normal, cause normally invisible information to become visible, and do many other things that can not be done in real, unsimulated situations. Teaching is widely held to be a cognitive skill (e.g., Leinhardt & Greeno, 1986), so it should be trainable with simulators as well. Training teachers on simulators should reap all the usual benefits that one gets with simulation-based training. For instance, as a teacher, I often wonder if what I said had the impact that I intended on a student. I would really like to see inside the student's head and note the changes in knowledge, if any. That can be done with a simulated student. If the intended change did not materialize, one can back up

the simulation, which erases the damage to the simulated student's knowledge, and try a different pedagogical approach. There is no way to do any of this with a real student. These are just some of the benefits of training teachers with simulated students. Unfortunately, most teacher-student interaction is conducted via conversations, and the proposed research explicitly avoids reactive instruction and feedback. The learning system that will be developed would need substantial but perhaps feasible modifications in order to be used in a simulation-based teacher trainer.

However, the proposed system could be used to evaluate written instructional materials. Anyone who has ever studied students trying to learn from written materials soon realizes how often they induce misunderstandings. For instance, most of the bugs found in subtraction can be blamed squarely on the design of the written materials used to teach subtraction (VanLehn, 1991). If a simulated student like Sierra (VanLehn, 1991) had been used to evaluate the material before publication, then these bugs might have been avoided. Similarly, Cascade should be usable for evaluating materials for teach physics and may help in finding misleading features.

Several studies of peer learning (e.g., O'Donnell et al., 1990; Webb, 1989) have shown that it is quite effective, but not all individuals benefit equally. For instance, students who explain things to other students often learn the most (Webb, 1989; pg. 29), but too many requests for explanation causes students to become taciturn and unhelpful. One possibility is to add simulated students to peer learning groups. The simulated students would take on various roles and exhibit just the right misunderstandings to cause human students to make explanations that will most help them, the human students, learn. Although this particular application will also require extension of Cascade so that it can "converse" with its peers, the possibilities of engineering a peer learning group so that the weaker student don't get left behind is exciting.

These are just three of the more mundane applications that a simulated student can be used for. Basically, once you understanding a phenomenon well enough to simulate it, then a whole world of possibilities opens up. But the first step, of course, is to reach that understanding. This brings us to a discussion of the approach to be taken by the proposed research.

3 The thesis of the proposed research

The central thesis of the proposed research is that most of the effects usually attributed to knowledge compilation are in fact due to three simple processes, all of which have independent motivation.

The first process is knowledge acquisition. While solving a problem (i.e., practicing), subjects often notice an opportunity to learn something about the task domain. For instance, if they reach an impasse, and they can not locate any mistakes in their path up to this point, then they often assume that they are missing some domain knowledge, and they set about to find out what it is. Sometimes subjects pause at the end of a problem and reflect on their solution, which allows them to see overall patterns. Episodes like these are sometimes found in protocols (VanLehn, 1991a; VanLehn & Jones, in press-b), and they indicate that knowledge acquisition can go on even when there is no explicit instruction and subjects are "just practicing." Needless to say, these acquisition events generate knowledge that can speed up problem solving, reduce incorrect solution, and transfer to later problems. Thus, knowledge acquisition during supposedly "pure" practice can explain some of the phenomena thought to be caused by chunking, schema/case acquisition or search tuning.

The second column of Table 2, which is labeled "Old Cascade," indicates which findings should be amenable to explanation via knowledge acquisition such as the kind found in the current version of the Cascade system. Table 2 uses the same notation as Table 1, so "++" means that the column's process explains the finding, "+" means that it partly explains it, and "?" means that it unclear

whether the process will be consistent or inconsistent with the finding. A blank means that this process is irrelevant to the finding. The last column indicates the hope-for coverage of a revised version of Cascade that includes all three processes.

The second process is a form of abstract planning that is particularly common in domains where the task is to analyze a physical situation mathematically. In such domains, the analyst must choose an abstraction or idealization of the physical system, then see if the mathematical treatment implied by that idealization will succeed in solving the problem. In many cases, the mathematical part of the analysis can be planned qualitatively. When the equations involved are simple (e.g., linear, or higher order with boundary conditions), knowing the values of all but one of the quantities in the equation suffices to determine the remaining quantity. Systems of equations can be similarly treated by counting the number of equations and the number of unknowns. In practice, these mathematical facts permit one to plan a solution. The solver might say, "Given the block's mass, I can get its weight via $w = mg$. From that and the tension on string 1, I can get the tension of string 2 via Newton's first law applied to the x-axis. From the tension in string 2, I can get the weight of the other block via Newton's law. And that's what the problem seeks." Such discussions are often observed in expert protocols (e.g., Larkin, 1983). Notice that the solver did not actually do the mathematical part of the analysis, but only planned it. Things could still go wrong. A variable might unexpectedly cancel out, for instance. Nonetheless, this qualitative planning phase is worth doing because it helps the analyst determine whether the choice of idealization (or "system" as the physicists call it) is going to work. For instance, in one protocol, Larkin (1983) observed an expert make 4 choices of system and rejecting the first three solely on the basis of a qualitative propagation like the one described above. Although this process is often observed, no one has named it yet, so let us call it *qualitative algebra.*

Qualitative algebra, when combined with system selection, can explain all the effects listed in Table 2 under the heading "solution plans." It is fairly clear from protocols of experts in Chi, Glaser and Rees (1982) that at least some experts do such an analysis before announcing their basic approach to the problem. It is also likely that experts use qualitative algebra when they classify problems, because it takes them 45 seconds to decide how to classify a problem which is 50% longer than novices take (Chi, Feltovitch & Glaser, 1981). The extra time is probably spent doing qualitative algebra. The rather complex and apparently contradictory findings about the influence of expertise on search strategy can, we think, be explained by assuming that practice increases the amount of qualitative algebra. Qualitative algebra can be conducted in either a forwards or backwards direction, but after it is done, the equations are generally produced in a forwards direction, as that reduces the writing because solving an equation can be folded into instantiating it. Depending on whether one is analyzing verbal protocols (e.g., Chi, Feltovitch & Glaser, 1981) , in which case one hears the qualitative algebra, or analyzing written protocols (e.g., Larkin, 1983), one sees different orderings for the equations produced. This explains why different investigators have found expert/novice strategy shifts. Although it is not worth going through the rest of the solution plan findings in Table 1, we believe that the combination of system selection and qualitative algebra can explain all of them. In particular, the geometric analog of qualitative algebra seems to be exactly what Koedinger and Anderson's (1990) experts are doing, so this hypothesis seems capable of explaining a finding that case-based reasoning and schema theory cannot easily explain.

The third process is a relatively passive memory, like the one assumed in the verbal learning tradition and its descendents. Two fundamental properties of such a memory are that (1) the more frequently an item is accessed, the easier it is to access again, and (2) the greater the similarity between the retrieval context and the encoding context, the easier it will be to access the item. Several of the long term learning effects can be explained by assuming that such a memory is

Finding	Old Cascade	Qualitative algebra	Memory	New Cascade
Basic effects				
Increases autonomy			+ +	+ +
Increase speed	+ +	+ +	+ +	+ +
Power law increase in speed			+ +	+ +
Increases accuracy	+ +	+ +	+ +	+ +
Decreases load			?	?
Increases retention		?	+ +	+ +
Facilitates acquisition	+			+ +
Practice never stops helping			+ +	+ +
Transfer				
Unit for predicting positive transfer	+ +			+ +
Asymmetric transfer	+ +			+ +
Short term negative transfer	+ +		+ +	+ +
Long term lack of negative transfer	+ +		+ +	+ +
Practice increases specificity of transfer			+ +	+ +
Solution plans				
Basic approach task		+ +		+ +
Classification of problems		+ +		+ +
Mentioning problem or solution types	+ +			+ +
Strategy differences		+ +		+ +
Estimation of difficulty		+ +		+ +
Self-monitoring		+ +		+ +
Second-order features/classification		+ +		+ +
Abstract planning		+ +		+ +
Episodic memory				
Memory for problems		+ +	+ +	+ +
Memory for solutions		+ +	+ +	+ +
Semantic memory				
Similarity tasks	+			+
Order in free-recall	?			?
Semantic priming	?			?
Miscellaneous				
Generalization can be deliberate	+ +			+ +
Slow increase in new rule usage			+ +	+ +
Incomplete memory for puzzle solutions			+ +	+ +
Conceptual problem solving improves	+ +		+ +	+ +

Table 2: A comparison of the proposed model to 30 empirical findings

used to store domain knowledge, intermediate results and other information in a model of problem solving. Newell and Rosenbloom (1981) have already shown that such a memory will explain the power law of learning, given various assumptions about the mixture of tasks and/or learning rates. It should be straight forward to explain how practice increases accuracy and retention. Decreases in cognitive load would require some assumptions about working memory, but probably only mild ones. In order to explain why practice decreases the subjects' reliance on external material and mnemonic tricks, we need to make the mild assumption that the subject first tries to find an item in memory before trying to find it in the textbook. This would explain why practice increases autonomy. The confusing interplay between short-term and long-term negative transfer is explained, somewhat unsatisfactorily perhaps, by an interplay of frequency and context effects on memory. In short, many of the classic findings in the practice and transfer literature can be explained with a simple model of memory that is amply motivated by experimental findings from the vast memory literature. For lack of a better term, we call such a model of memory a *black box memory* because it really doesn't explain the frequency and context effects, but simply reproduces them for the benefit of the surrounding Cascade system.

Although we believe that almost all of the findings often attributed to knowledge compilation mechanisms can be explain by knowledge acquisition, qualitative algebra and a black box memory, we equally certain that some kind of episodic memory for problem solving is also available. This is clear from introspection as well as from the comments from the subjects, such as "this is where I got stuck last time." Perhaps these episodes are stored in some kind of case-based memory or one of the other classic knowledge compilation memories. However, the impression one gets from several years of working with protocols of people learning is that these kinds of remindings, while certainly available, make little difference in the person's problem solving.[1] Basically, knowing that something happened at a similar point last time doesn't help you do anything different this time unless you've either learned something relevant in the meantime or were particularly stupid last time.

4 Research plan

There are four major pieces of work to be performed: (1) Build an model that integrates knowledge acquisition and compilation. (2) Evaluate it against the known empirical findings. (3) Produce protocols of long-term learning in a controlled situation. (4) Evaluate the completeness of the model by fitting it to the protocol data. These tasks will be discussed in separate subsections. However, tasks 3 and 4 will only be tackled if the length of funding is extended from the proposed 1 year to a total of three years.

4.1 Augmenting Cascade

The cognitive model to be built will be an extension of our current cognitive model, Cascade. Cascade models knowledge acquisition but not knowledge compilation. Domain knowledge is represented with small units, about the size of production rules.[2] Problems are solved and examples are explained by backwards chaining with these rules. Cascade also has two forms of analogical

[1] In a more memory intensive skill, such as learning a text editor with lots of commands, remindings might be more powerful (Ross, 1984; 1989).

[2] Cascade3 uses conditioned equations, which are described in VanLehn, Jones & Chi, in press. Cascade5 uses ordinary if-then rules (Horn clauses), similar to the ones used for exposition in VanLehn and Jones, in press-b. The conditioned equations facilitate learning, but the if-then rules are more general. We are currently working on a way of combining the advantages of the two representations.

problem solving. One, called *transformational analogy* models the subjects' equation hunting. Cascade finds a problem analogous to the current one, and searches through its solution steps to find an equation that it believes will help it achieve its current goal. The second form of analogical problem solving, called *analogical search control*, also begins by finding a problem that it analogous to the current problem. However, it seeks advice on which rule to apply to the current goal by finding an old goal that is analogous to its current one. In the current version of Cascade, neither form of analogy produces new rules. However, knowledge acquisition does occur when Cascade cannot achieve a goal by any of the regular means. At such an impasse, it tries to achieve the stuck goal by using rules that are known to be overly general and probably wrong. If one of these rules succeeds in achieving the goal and this allows Cascade to successfully complete the problem, then an instantiation of the overly general rules is created and added to Cascade's list of domain rules. This learning technique is called *explanation-based learning of correctness*, or EBLC.

Although Cascade is rather simple, its behavior matches subjects' behavior quite closely. The model was fit to protocols from 9 subjects who learned physics by studying examples and solving problems (VanLehn & Jones, in press-b). Over 95% of Cascade's inferences were also made by the subjects, and between 60% and 90% of the subjects' inferences were made by Cascade.

In order to make Cascade an adequate model of knowledge compilation as well as knowledge acquisition, four modifications are needed. The first modification is to make Cascade distinguish between saving intermediate results in working memory and saving them by writing them on the worksheet. This modification is necessary in order to facilitate comparison of Cascade's behavior with the subject's. Without it, it will be difficult to demonstrate, for instance, that practice decreases working memory load. More importantly, the data collection system to be used in the long-term learning study is essentially an electronic worksheet, so it can record all the subject's writing and erasing actions. Although we plan to tape the subjects' think-aloud protocols, we may be able to avoid transcribing these tapes if the electronic/written record is rich enough. Thus, comparison of the subjects' behavior with Cascade would be facilitated if Cascade also make explicit writing actions.

The second modification to Cascade is to add a black box memory. We have already worked out where the memory accesses should go. Table 3, which is taken from VanLehn, Jones and Chi (in press), uses italics to indicate where in the basic Cascade interpreter the black box memory should go. It also suggests reconstruction and backup strategies that would be appropriate when memory retrievals fail.

Cascade is intended to be used both to generate behavior and to analyze behavior generated by subjects. When used to generated behavior, the black box memory has to be parameterized with appropriate functions for calculating retrieval probabilities based on access frequency and context. Context has to be represented somehow, perhaps as a long feature vector extracted from the current problem description, any intermediate results written so far and the current goals. The details of the black box memory only matter for a few findings, such as the ones involving negative transfer. We don't plan to spend much time on trying to get the frequency and context effects right. The major test of Cascade will come when it is fit to subject behavior generated during the long-term learning study. Here we should sometimes be able to infer the memory accesses of the subjects, and these could perhaps be used to test whether they show the usual frequency and context effects.

The third modification to Cascade is to add a model of qualitative algebra. This is rather simple, and has be partially implemented already. The basic trick is to first solve the problem with a "use qualitative algebra" flag turned on. This causes all equations to be treated as sets of variables, and the usual algebraic manipulations are replaced by simple known/unknown counting. Cascade's analogical search control mechanism uses the results of this derivation to guide it during a second pass at the problem with the flag turned off. Preliminary experience suggests that this

In problem P, to find a value V for quantity G or to show that V is the value of quantity G, try these methods in order until one succeeds:

1. Analogical search control.
   Do the following five steps in order, failing if any one fails and the failure can't be handled:
   a. Retrieve an example E that is similar to P.
      If retrieval fails, then
      flip pages looking for an example with a diagram that is similar to P's diagram.
   b. Retrieve a mapping between E and P.
      If retrieval fails, then
      reread problem statements of E and P, and
      create a mapping.
   c. Using the mapping, substitute terms in G to form a target goal T.
   d. Retrieve a triple (E T R), where R is bound by retrieval to a rule.
      If retrieval fails, then
      reread lines of E's solution to stimulate recall.
      If rereading lines stimulates only partial recall, then
      redo the derivation of the line that stimulated partial recall, and
      retrieve a triple from the new derivation.
      If rereading lines fails to stimulate recall, then
      redo the whole derivation, and
      retrieve a triple from the new derivation.
   e. Show that R's conditions are met.
   f. Apply R's equation to G and V.
   g. Create a triple (P G R).
   h. Return whatever Step f returned.

2. ~~Rule-KB search:~~ Regular rule selection and application
   Do the following steps in order, failing if any one fails and the failure cannot be handled:
   a. Retrieve a domain rule (or any rule if this is not pass 1)
      whose equation contains a quantity unifying with G and
      whose condition is met by the current situation. Call the rule R.
   b. Plant a backup point so that a different rule can be
      retrieved if R leads to failure.
   c. Apply R to G and V.
   d. If R is an overly general rule, then
      create a specific version of the rule by instantiating R
      and substituting variables for problem-specific constants.
      Call this new rule R and
      mark it as a domain rule of P's task domain.
   e. Create a triple (P G R).
   f. Return whatever Step e returned.

(continued)

Table 3: The main loop of Cascade's interpreter, showing memory locations

3. Transformational analogy.
   If a problem is being solved, then
   do the following steps in order, failing if any one fails and the failure cannot be handled:
   a. Retrieve an example (as in Step 1a).
   b. Retrieve a map (as in Step 1b).
   c. Create a target goal T via mapping G (as in Step 1c).
   d. Retrieve a line of the example that contains T.
      If retrieval fails, then reread each line to see if it contains T.
   e. Substitute terms in the line via the map to put it in terms of P.
   f. Apply the line's equation to G.
   g. Return whatever Step f returned.

4. Analogy abduction.
   If this is the third pass, and
   an example is being explained and a value V for G is known, then
   do the following steps in order, failing if any one fails and the failure cannot be handled:
   a. Create an analogy rule R (see text), and
   b. Mark it ~~with~~ as a domain rule of P's task domain, and
   c. Create a triple (P G R).
   d. Return success.

5. Impasse: No rules apply to G.
   If there are backup points, then resume one,
   else if this is Pass 1, then start over with Pass 2,
   else if this is Pass 2 and an example is being explained, then start over with Pass 3,
   else fail utterly. This problem/example cannot be solved/explained.

To apply an equation E to a quantity G when the value is unknown:
1. Let S be all quantities in E except G.
2. Recurse to find the values of each quantity in S.
3. Substitute values for quantities in E.
4. Solve E for G.
5. Return the result as G's value.

To apply an equation E to a quantity G when the value V is given:
1. Solve E for G, obtaining expression X.
2. Match X to V, obtaining a set S of quantity-value pairs.
3. Recurse to show that each quantity in S has the value with which it is paired.
4. Return success.

The main loop of Cascade's interpreter, continued.

simple mechanism may adequately simulate qualitative algebra.

Lastly, our recent fitting of Cascade to protocol data (VanLehn & Jones, in press-b) indicated that Cascade's model of analogical problem solving was too simple. In particular, people learn from doing analogical problem solving but the current version of Cascade does not. This, and other known problems with the Cascade model of knowledge acquisition must be fixed.

4.2 Explaining the 30 long-term learning effects

The basic objective of this task is to perform simulations runs for most of the findings listed in Table 2 and demonstrate that Cascade can account for them. Whenever possible, if the literature reports quantitative findings, such as the power law or the production-overlap prediction of transfer, we will try to generate numerical predictions using Cascade that fit the quantitative regularity. In some cases, however, we will have to settle for monotone relations, such as showing that experts are more accurate at estimation than novices.

For the purposes of this study, a number of modules will have to be added to Cascade. These modules use the same rules as Cascade's problem solver, but they perform other tasks, such as classifying problems or estimating their difficulty. Building one of these modules is a nice small project, suitable as an exercise for new graduate students.

Whether Cascade will actually be able to explain all the findings is of course still uncertain. We may have to make significant changes to Cascade in order to accommodate some of them.

4.3 The long-term learning study

This project, and the field in general, needs to see what really happens over long periods of practice in a rich problem solving domain. Our hypothesis is that most changes in behavior are due to deliberate knowledge acquisition by the subject, as well as a growth in the frequency of qualitative algebra and some standard memory effects. This hypothesis grew out of our inability to see in protocol data any of the postulated knowledge compilation effects. Although one could design a series of traditional experiments to test our hypotheses, we prefer to conduct one extended protocol-taking experiment. Partly this is because that is the methodology that we are most experienced in, and partly it is because we wish stay close to the kind of observable behavior that teachers and others must deal with on a daily basis. That is, we are more concerned with explaining, say, the increasing autonomy of learners than the changes in their performance on semantic priming tasks because one can see increasing autonomy in protocols and in real life, but semantic priming is virtually invisible.

The study, in a nutshell, is to take protocols of learners as they move from novicehood to expertise in a small area of physics. As explained below, we think that it will take 100 to 200 hours of training to make the transition. The length of this study presents some problems as well as some opportunities. The main problem is coping with that much protocol data. This we will handle by using an on-line assessment system, Olae (VanLehn et al., in prep), that is being developed with existing ONR funding. It will handled much of the data collection and analysis. Moreover, Olae does more than just collect non-verbal protocols. It can administer several classic expert-novice tasks and analyze the results.

The richness of this data and the anticipated length of the training afford many interesting opportunities. We would like to introduce some within-subject manipulations in the training material in order to tests specific hypotheses. Given the length of the training, several such manipulations may be possible, but they would have to be carefully designed. This section sketches some ideas for potential manipulations, but the design is still very much in flux. Needless to say, we will

consult our more experimentally oriented colleagues at LRDC (Micki Chi and Walter Schneider, in particular) who have experience in designing and conducting large-scale training and protocol studies.

The subjects for the main study will be undergraduates with strong backgrounds in mathematics. We anticipate starting about 6 subjects, then dismissing half of them after the second week and keeping only those who give the clearest protocols and show the most signs of being willing to complete the training.

The subject matter will be the subset of classical, Newtonian mechanics that deals with force and energy in mechanical systems with no collisions and no rotation. This subset of mechanics is covered in chapters 5 through 9 of Halliday and Resnick (1981), a standard college physics textbook. In the scientist-and-engineer track of physics taught here at Pitt, these chapters occupy the middle two months of a four month semester.

The training to be given to the subjects will mimic the kind of self-studying that many scientists and engineers rely on. The subjects will read the textbook chapters, but their studying of examples and solving of problems will be done on the Olae system. In previous studies of this kind, including the 1989 and 1982 protocols collected by Chi's group, subjects were not supposed to be given any feedback on the correctness of their answers. This was intended to make analysis of their learning easier. However, the experimenters sometimes gave the subjects feedback, and it is clear that these slips were very important for increasing the subject's learning. Therefore, we are contemplating adding a limited amount of feedback to the training in order to increase the subjects' learning rate. One type of feedback would be to allow the subjects to see the correct answer to a problem after generating their own. This is ecologically valid, as it is the same kind of feedback available in some physics books, which print answers to selected exercises in the back of the book. We will run pilot subjects in order to understand the implications of various feedback regimes.

The training will be delivered on the Olae system, which will be modified to deliver whatever feedback we decide upon. In addition to monitoring the students' problem solving, by keeping track of what they draw and the equations they enter, Olae also keeps track of which examples they refer to and where they look when refer to an example. Olae also administers (1) a problem classification task similar to the one used by Chi, Feltovitch and Glaser (1981), (2) a task which asks subjects to describe their basic approach to a problem and estimate its difficulty, and (3) conceptual physics tasks, similar to those used by Halloun & Hestenes (1985), McClosky et al. (1983) and many others. Olae's data collection section is in good shape. As of this writing, 5 pilot subjects and 5 real subjects have been run. The user interfaces appear to be working unobtrusively, as they should.

For the purposes of this study, we would like augment Olae in several ways. First, we want to add some kind of user interface that will allow us to track a person doing qualitative algebra. We're not sure how to do this yet. Second, we are contemplating adding a semantic memory task designed by Tim Goldsmith and Pedar Johnson, and reported at the fall 1991 ONR contractors meeting in Pittsburgh. We may also add a Chase and Simon episodic memory task. In both cases, these tasks will be pilot tested with pencil and paper versions before adding them to Olae, as it is not clear that they will translate well into physics. There is some interest in adding these tasks to Olae anyway, as they will, we hope, improve its ability to assess expertise. These enhancements will allow us to observe, within a single study, almost all the long-term learning effects. The findings that cannot be observed are: self-monitoring, free recall, semantic priming and memory for puzzle solutions.

We are not yet certain how much training will be enough to get significant expert-novices contrasts. The 1989 Chi experiments had subjects study one chapter for 5 hours, and subjects were still a long way from being experts even on that one small piece of subject matter. The

1982 studies trained subjects for 16 hours on 2 chapters, and they were also not at all expert. We estimate that it will take about 200 hours to see expertise begin to develop, or 40 hours per chapter. This is about three times what students in a typical Pitt class spend on the same material. It corresponds roughly to the experience graduate students in physics get who are trained in classical mechanics as freshman, juniors and first-year graduate students. Each time they take it, they use more sophisticated mathematical tools. Our subjects will always use the same mathematical tools (algebra, but no calculus), so their training in classical mechanics is only partially equivalent to a graduate student's.

The expository training materials will be taken from Halliday and Resnick, whose text has gradually improved over the years. The most recent edition is actually quite good, and shows clear signs that the authors have been reading the cognitive science literature. The examples and problems will be taken from a corpus of about 3000 problems that we have collected from various physics textbooks. The materials will be pretested by having Cascade learn from them. This tends to uncover quirks in problems. Moreover, Olae's analytic database is built by running Cascade and keeping a trace of its reasoning, so having Cascade solve the problems before the subjects' do is necessary in any case.

A major design decision that we have not yet made is the exact selection of problems, examples and their sequence. The design requires some thought, for there are several interesting opportunities. For instance, one design would cross surface with deep features, and draw problems from some of the cells much more frequently than from the others. This differential training could produce interpretable effects on the subjects' performance on the assessment tasks. For instance, it would be interesting if heavy training on force problems enhanced the subject's episodic memory for force problems but not energy problems.

4.4 Data analysis and simulation

The fourth and last task to be accomplished is analyze the data produced during the long-term learning study. We anticipate conducting three types of analysis.

The first analysis is the one conducted by Olae itself. Essentially, it uses a Bayesian network to fit Cascade to the subject's behavior. The network associates a probability with each Cascade rule. The probability indicates the posterior probability that the subject has the rule given that the subject generated the observed behavior. A good fit corresponds to all the rules having values that are either near 1.0 or near 0. A poor fit has many rules whose probabilities of possession are near 0.5. The chief value of the Bayesian fit for this study is that it indicates sections of the protocol where the subject is behaving with unexpected competent or incompetence. These sections of the protocol deserved a closer look. We intend to record verbal protocols throughout the study, but only transcribe and analyze those sections of the protocol that the Bayesian analysis indicates are worth exploring.

A second group of analyses will attempt to test the thesis of this research, which is that long term learning effects can be explained by knowledge acquisition, qualitative algebra and a black box memory. One analysis, for instance, could examine the subject's actions to see how the subject is wasting time when a novice, and how that waste is detected and reduced. For instance, it is fairly easy to tell when a subject has searched unsuccessfully, and it may be possible to tell when the subject has used qualitative algebra. Given data like these, we can tell whether the speed up in problem solving is due most to reduced search or to increased use of qualitative algebra, or perhaps to that mysterious reduction in waffling mentioned earlier.

A third group of analyses will be less theoretically motivated. Because we aim to collect multiple measures of expertise, it will be interesting to see if they "correlate." That is, if a person's perfor-

mance has become almost automatic for a certain class of problems, what will their performance on problem classification or episodic memory be like on that same class of tasks? Perhaps all the various measures of expertise should correlate temporally and topically, but it would be interesting if some didn't.

5 Schedule

The first year would be occupied with augmenting Cascade (task 1) and designing the materials for the long term learning study. The second year would involve demonstrating that Cascade can model most of the long-term learning effects (task 2) and in running the long-term learning study (task 3). The third year would be devoted to analyzing the data from the long-term learning study.

6 Bibliography

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

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

Anderson, J.R. (1989). The analogical origins of errors in problem solving. In D. Klahr & K. Kotovosky (Eds.) *Complex information processing: The impact of Herbert A. Simon.* Hillsdale, NJ: Erlbaum.

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

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

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

Anzai, Y., & Simon, H. A. (1979). The theory of learning by doing. *Psychological Review*, 86 (2), 124-140.

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

Bloom, B. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, June/July, 4-16.

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

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

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

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

Chi, M. T. H., Feltovitch, P. J & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5, 121-152.

Chi, M. T. H., Glaser, R. & Rees, E. (1982). Expertise in problem solving. In R. J. Sternberg (Ed.),*Advances in the Psychology of Human Intelligence* Vol. 1, pp. 7-75. Hillsdale, NJ: Erlbaum.

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

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

Crossman, E.R.F.W. (1959). A theory of the acquisition of speed-skill. *Ergonomics, 2,* 153-166.

Elio, R. & Scharf, P. B. (1990). Modeling novice-to-experts in problem-solving strategy and knowledge organization. *Cognitive Science.* 14, 570-639.

Farr, M.J. (1987). *The Long-term Retention of Knowledge and Skills.* New York: Springer-Verlag.

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

Fitts, P.M. & Posner, M.I. (1967) *Human Performance.* Belmont, CA: Brooks Cole.

Halliday, D., & Resnick, R. (1981). *Fundamentals of Physics.* Second edition. New York: Wiley.

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

Hall, R. J. (1988). Learning by failing to explain: Using partial explanation to learn in incomplete or intractable domains. *Machine Learning,* 3, 45-77.

Jones, R. M. (in press). Problem solving via analogical retrieval and analogical search control. In S. Chipman & A. Meyrowitz (Eds.), *Machine Learning: Induction, Analogy, and Discovery.* Boston: Kluwer Academic.

Jones, R. M. (1989) *A model of retrieval in problem solving.* Doctoral dissertation. Information and Computer Science, University of California, Irvine.

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

Kambhampati, S. (1990). A theory of plan modification. *Proceeding of the Eighth National Conference on Artificial Intelligence.*(pp. 176–182). Menlo Park, CA: AAAI Press/MIT Press.

Koedinger, K.R. & Anderson, J.R. (1990) Abstract planning and perceptual chunks: Elements of expertise in geometry. *Cognitive Science, 14,* 551-578.

Kuhn, D., Amsel, E., & O'Loughlin, M. (1988). *The development of scientific thinking skills.* Orlando, FL: Academic Press.

Langley, P. W. (1983). Learning effective search heuristics. *Proceedings of the Eighth IJCAI,* Karlsruhe, W. Germany, pp. 465-68.

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

Leinhardt, G., & Greeno, J. The cognitive skill of teaching. *Journal of Educational Psychology,* 78, 75-95.

Lewis, R. L., Newell, A. & Polk, T. A. (1989). Toward a Soar theory of taking instructions for immediate reasoning tasks. *Proceedings of the Annual Conference of the Cognitive Science Society*

McCloskey, M. , Washburn, A., & Felch, L. (1983). Intuitive physics: The straight-down belief and its origin. *Journal of Experimental Psychology: Learning, Memory and Cognition,* 9, 636-649.

Michell, T. M, Utgoff, P. E. & Banerji, R. B. (1983). Learning by experimentation: Acquiring and refining problem-solving heuristics, In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach,* Palo Alto, CA: Tioga.

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

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

Newell, A. & Rosenbloom, P.S. (1981) Mechanisms of skill acquisition and the law of practice. In J.R. Anderson (Ed.) *Cognitive Skills and their Acquisition.* Hillsdale, NJ: Erlbaum.

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

Donnell, A. M., Dansereau, D. F., Hall, R. F., Skaggs, L. P., Hythecker, V. I, Peel, J. L. & Rewey, K.L. (1990). Learning concrete procedures: Effects of processing strategies and cooperative learning. *Journal of Educational Psychology*, 82 (1), 171-177.

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

Rivest, R. & Sloan, R. (1988). Learning complicated concepts reliably and usefully. *Proceedings of AAAI-88*, Los Altos, CA: Morgan-Kaufman.

Ross, B. H. (1984). Reminding and their effect in learning a cognitve skill. *Cognitve Psychology*, 16, 371-416.

Ross, B. H. (1989). Remindings in learning and instruction. In S. Vosniadou & A. Ortony (Eds.), *Similarity and Analogical Reasoning*, pp. 438-450. Cambridge, MA: Cambridge University Press.

Samuel, A. L. (1963). Some studies in machine learning using the game of checkers. In E. A. Feigenbaum & J. Feldman (Eds.) *Computers and Thought*, New York: McGraw Hill.

Shiffrin, R.M. & Dumais, S.T. (1981). The development of automatism. In J.R. Anderson (Ed.) *Cognitive Skills and their Acquisition.* Hillsdale, NJ: Erlbaum.

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

Singley, M.K. & Anderson, J.R. (1989). *The Transfer of Cognitive Skill.* Cambridge, MA: Harvard.

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

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

VanLehn, K. (1989). Problem solving and cognitive skill acquisition. In M.I. Posner (Ed.) *Foundations of Cognitive Science.* Cambridge, MA: MIT press.

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

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

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

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

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

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

VanLehn, K. & Jones, R.M. (in press-b). Learning by explaining examples to oneself: A computational model. In A. Meyrowitz & S. Chipman (Eds.) *Cognitive Models of Complex Learning.* Boston: Kluwer Academic Publishers.

VanLehn, K., Martin, J., Chi, M. T. H., Glaser, R. G., Rubin, J. & Ur, S. (in prep.) *Integrating non-standard tasks with diagnostic monitoring to assess expertise.*

Winston, P. H. (1975). Learning structural descriptions from examples. In P. H. Winston, (Ed.), em The Psychology of Computer Vision. New York: McGraw-Hill.

Webb, N. M. (1989). Peer interaction and learning in small groups. *International Journal of Educational Research.* 13, 21-40.