Applications of Simulated Students: An Exploration

Kurt VanLehn
Stellan Ohlsson
Learning Research and Development Center
University of Pittsburgh, and
Rod Nason
Centre for Mathematics and Science Education
Queensland University of Technology

December 16, 1993

Abstract

It is now possible to build machine learning systems whose behavior is consistent with data from human students. How can education use such simulated students? Applications that help three user groups are discussed. *Teachers* can practice the art of tutoring by having them teach a simulated student. Using a simulation instead of a real student allows teachers to see how their actions affect that student's knowledge, to undo their actions, and to try their skills on students with varying prior knowledge and learning strategies. *Students* can learn in collaboration with a simulated student. Because the simulated student can be simultaneously an expert and a co-learner, it can scaffold and guide the human's learning in subtle ways. *Instructional developers* can test their instruction on simulated students. Unlike formative evaluations with real students, a simulation-based evaluation can indicate exactly what piece of the instruction caused which pieces of knowledge, and thus help developers troubleshoot their instructional designs early in the design process. For each of these three areas of application, inherent technical limitations, existing systems and prospective systems are discussed.

New technologies have made it possible for computer systems to learn sophisticated skills and knowledge. Both symbolic machine learners and artificial neural networks have demonstrated remarkable successes, especially within the last decade. During that time, empirical work on human learning of higher level skills and knowledge has expanded enormously (see VanLehn, 1989a, for a brief review), and several general theories of skill acquisition have been proposed (Anderson, 1983; Holland, Holyoak, Nisbett & Thagard, 1986; Newell, 1990; Ohlsson, 1993; VanLehn, 1989b). It is

now possible to build computer systems that simulate human students as they learn educationally significant subject matter such as arithmetic (Badre, 1972; Ohlsson & Rees, 1991; VanLehn, 1987, 1989b), algebraic equation solving (Neves, 1978, 1981), geometry (Anderson, Greeno, Kline & Neves, 1981; Neves & Anderson, 1981), college physics (Elio & Scharf, 1990; Larkin, 1981; VanLehn, Chi & Jones, 1991), organic chemistry (Ohlsson, 1993), automobile repair (Redmond, 1989, 1992), electronics (Mayer, 1990) and Lisp programming (Anderson & Thompson, 1989; Pirolli, 1991).

A technology that can simulate students presents an opportunity for education. How best can education take advantage of this new technology? We suggest that there are three broad classes of opportunities. *Teachers* can develop and practice their craft on simulated students. *Students* can work collaboratively with a simulated peer or by teaching a simulated student who is less knowledgeable than themselves. *Instructional designers* can pilot test their products on simulated students in order to get precise feedback early in the design process.

This paper explores each of these three application areas. Most journal articles present new work or critically review a body of existing work. However, our work is just beginning in these areas, so we can report mostly on design constraints that have been uncovered during the early stages of development. Although others have produced educational systems based on machine learning models of students, the body of work is as yet too small to support an integrated or critical review. Nonetheless, even this modest amount of designing and prototyping has revealed some non-obvious conclusions. In particular, we will argue that teacher training is currently the most feasible application of simulated students. Development of the other two application areas, collaborative learners and formative evaluations, require overcoming significant challenges in human-computer interaction and machine learning, respectively. However, this and our other conclusions are tentative as they are based only on a small body of work and some preliminary designs. Our overall conclusion, and the main motivation for writing this paper, is that the technology for cognitive simulation of learning is now ripe for application. The educational problems exist and the simulation technology exists. It is just a matter of putting them together. This article is a speculative mixture of optimism and caution—an exploration, just as the title says.

Each application area will be discussed in separate sections. A major consideration will be limitations due to the current state of the simulation art. The few prototype systems that exist will be reviewed briefly. In order to make concrete our vision of the future, we sometimes offer

speculative scenarios of humans interacting with a hypothetical system. However, before embarking on the exploration of the three application areas, it is worth clarifying the basic notion of a simulated student and introducing two important distinctions among simulated students.

1 Simulated students are models of human learning

In order to model a single episode of learning, the simulated student is given two inputs: (a) a formal representation of the relevant parts of the human student's knowledge just prior to the episode, and (b) a formal representation of the instruction given to the human student. The model produces two outputs: (a) a formal representation of the human student's knowledge just after the instruction, and (b) a formal representation of the student's behavior during the learning episode.

Simulations of longer periods of learning are accomplished by running many cycles of the above process. Interaction with the human student is formalized as a sequence of instructional units. These units might be as large as a whole lesson or as small as a single proposition in a conversation. Each unit is given to the simulated student, which produces behavior and modifications to the knowledge base on each cycle. These behaviors and modifications should match the human student's behaviors and knowledge shifts, in so far as they can be determined.

Many kinds of simulated students exist. It is helpful to classify them along two dimensions: the granularity of their knowledge and processes, and the extent to which their model of learning is table-driven. These are not the most common distinctions used to classify expert and student models (Anderson, 1988; Dillenbourg & Self, 1992a; VanLehn, 1988), so the next two sections will discuss them in some detail.

1.1 The grain-size of a simulated student

The grain-size or granularity of a simulated student refers to the amount of detail in its knowledge representations and processing. A fine-grained model of physics learning might have knowledge represented by the rules shown in Table 1. Its problem solving and learning process would also be quite involved with getting the details right. Such fine-grained processing allows its predictions about observable behavior to be quite detailed. For instance, it could predict the order in which vector equations are written on the worksheet. As a second example, Ohlsson (1987) and Ohlsson, Ernst and Rees (1992) describe a model in which individual eye movements are represented – a

Table 1: An example of fine-grained knowledge of physics

```
1.  If B is a body,
       S is a taut string, rope or chain,
       S is tied to B,
    then there is a tension force T on B due to S.

2.  If there is a tension force T on B due to S,
       and S is inclined at an angle of A degrees,
    then T is inclined at an angle of A degrees.

3.  If there is a tension force T on B due to S,
    then magnitude(T)=tension(S).
```

very fine-grained level of analysis. The grain-size distinction applies to the whole simulated student: knowledge, processes and behavior.

In contrast, a large-grained model of physics might represent knowledge with atomic features, such as

```
knows-kinematics-of-uniform-acceleration

knows-Newton's-laws-of-motion

knows-gravitational-force-law

knows-elementary-contact-force-laws

knows-conservation-of-energy

knows-definition-of-kinetic-energy

knows-definition-of-potential-energy
```

For this simulated student, a textbook chapter of instruction could be formalized by listing the number of times each concept was referenced, either in the text or in the exercise problem solutions. The learning process could be formalized as a neural network or Bayesian network that changes the weights of the concepts in the knowledge base as a function of the frequency of occurrence of the concepts in the instruction. Such a simulated student would be fairly large-grained and therefore only a rough approximation to a human learner.

The two grain sizes exemplified represent the end points of the granularity scale. Simulated students can be located between these extremes.

1.2 Tabular vs. algorithmic simulations of learning

Another major distinction among simulated students is whether the processes that model learning are primarily tabular or primarily procedural. In principle, each cycle of a simulated student computes a function whose inputs are the prior knowledge and the instruction, and whose outputs are the learning-modified knowledge and the behavior. This function can be implemented as a table or an algorithm (procedure). For instance in the large-grained model mentioned earlier, it would be quite feasible to assign new weights to concepts by looking them up in a table indexed by their old weights and their frequency in the instruction. On the other hand, the new weights could also be computed by an algorithm.

One advantage of tabular models is that they can be implemented without really understanding the human learning processes, given that one has enough data from human learners to fill the cells of the tables. Another advantage is that tabular models of learning can be more accurate than algorithmic models because they usually have more degrees of freedom for fitting the data. One disadvantage is that tabular models are only feasible when the knowledge and instruction can be broken up into pieces that can be used to index tables. This may not do justice to important aspects of the human learner's understanding. Another disadvantage is that a tabular model can accept a piece of instruction only when it is already in the model's table. Although a tabular model can handle novel sequences or combinations of instructional pieces, it cannot handle new pieces.

Neural and Bayesian networks are partly tabular and partly algorithmic. Their structure imposes some theoretical constraints on the relationship between inputs and outputs, but they can still be parameterized from human learning data without really having a well-understood theory of that data. Whether a simulated student is tabular or algorithmic is not a binary distinction, but a matter of degree.

Although there are certainly other important distinctions among simulated students (c.f. Dillenbourg & Self, 1992a; VanLehn, 1988), the granularity and the tabular/algorithmic dimensions are sufficient for the subsequent discussions.

2 Simulation-based teacher training

Simulated students can be used to train teachers by having the teachers teach the simulated students and observe whether their teaching has the desired effects. This idea is not new (Doak & Keith, 1986). However, because simulations of learning have only recently become available, most existing systems train teachers in activities that do not directly involve students learning. Simulation-based systems exist that train teachers to diagnose arithmetic bugs (Brown & Burton, 1978; De Corte, Verschaffel & Schrooten, 1991; Nason, 1991), to organize reading groups (Shelley & Sibert, 1991) and to manage classroom motivation (Wood, 1991). Now that the technology is available for simulating learning, simulations can help train teachers in activities that directly involve students learning. However, there are still some limitations imposed by the technology.

2.1 Limitations of simulation-based teacher training

Three limitations spring to mind. First, teachers must often work with a whole classroom of students, but most of the current cognitive simulations of learning (op. cit.) assume that the student is either working alone or under the supervision of an individual tutor. There are as yet no models of students learning in groups or in classroom settings. Although machine learning has begun to explore multi-agent learning and cognitive psychologists have begun to study learning in groups, it may take a decade or more before we have good simulated students for multi-student learning situations. Because the current state of the simulation art allows us to simulate only students working alone or with a tutor, near-term applications should focus on enhancing teachers' tutoring skills.

A second limitation is that current technology for speech and text processing is very limited, and yet natural language communication is an important part of any human-human tutoring session. This is, of course, a familiar problem for the developers of intelligent tutoring systems, where the machine plays the role of teacher instead of student. The ITS community has found that a fairly natural communication can occur without natural language processing by using menus, graphics and limited linguistic processing. However, these techniques still work best only with certain subject matters, such as mathematics, where even human-human communication is mostly via a formal language. These same limitations apply to simulation-based tutor training systems as well as ITSs. Natural language processing has been making steady progress for many decades, but it

will probably require several more decades before simulated students can actually converse with the teacher in natural language.

A third limitation is that tutoring can have a profound effect on the student's motivation and self-confidence (Lepper, Aspinwall, Mumme & Chabay, 1990), but none of the existing simulated students model such effects. Fortunately, computational models of affect are beginning to appear (e.g., Ortony, Clore & Collins, 1988). It will probably be only a few years until we see simulated students getting frustrated and depressed when they cannot solve problems that they think should be easy, and becoming elated, excited and self-confident when they solve problems they think are hard. Until then, simulated students can only model students whose maturity and commitment is so robust that their self-confidence and motivation are not much affected by tutoring.

Given these limitations, the current technology for modeling learning should be most useful in training tutors of highly motivated students learning formal subject matters, such as science, mathematics, engineering, equipment troubleshooting, and so on.

2.2 Advantages of tutor training systems

Students typically spend much more time in the classroom than in tutoring sessions, so is it really worth while to improve teachers' tutoring skills? This is a difficult policy question whose full discussion would go far beyond the scope of this article. However, because time with a tutor is so rare, it is important that the time be used as effectively as possible. Tutors vary widely in their effectiveness. Some teachers, even very experienced ones, treat a tutoring session just like a classroom and make only minor adjustments in their curriculum scripts (Putnam, 1987). On the other hand, professional tutors exhibit almost miraculous skill, converting remedial math students into highly motivated, top-performing students (Lepper, personal communication, 1992). If schools can afford to tutor students for only a short time, it is important that the tutors be skilled at tutoring.

It is important that future teachers and tutors understand is that their role is not necessarily to "tell" things to their pupils, but rather to shape the tutorial situation so that students will construct their own knowledge. A tutor-trainer could allow teachers to discover for themselves how important it is to allow students to construct their own knowledge. A scenario presented later demonstrates how this constructivist philosophy can be employed to teach itself.

Tutoring, like all cognitive skills, requires practice. Because tutors must react almost instantaneously to the student, they must practice enough to make their responses nearly automatic. Automatizing even a single rule for a cognitive skill takes hundreds of trials of practice (Carlson, Sulivan & Schneider, 1989). Worse yet, novice tutors are not a blank slate, but often have bad habits that are difficult to break (Swanson, 1992). These features suggest that becoming an expert tutor might take much more practice than becoming an expert in other cognitive skills. Clearly, a simulation-based practice environment would make logging large numbers of practice hours much easier.

Simulation-based practice environments have an excellent track record in training complex skills such as flying an airplane, tactical planning, troubleshooting, air traffic control, financial decision making, and many others. It is not so farfetched to use simulations in training yet another professional skill, teaching. The traditional advantages of simulation-based training should also apply to teachers learning to tutor. In particular:

- Teachers can inspect the simulated student's knowledge base, and thus tell if their tutoring actions are having the intended effects.
- If the teachers do not like the effects of their most recent actions, they can reset the simulated student's knowledge to an earlier state and try again.
- Teachers can replay a tutorial session, in order to study its effects on the simulated student's knowledge.
- When teachers work in teams with a simulated student, they can discuss their theories of what the student is learning and how best to teach it without any danger that the student will "overhear" their discussion or get bored and leave if the discussion goes on for a long time.
- Teachers can modify the simulated student's initial knowledge, then replay the tutorial session in order to see how different students would react to the same treatment.
- Teachers can experiment ethically with novel teaching tactics. If the tactic fails, it will not hurt any human students.

- As in Sherlock (Lesgold, Lajoie, Bunzo & Eggan, 1991), simulated students can be rigged to make important situations occur much more frequently than they would with human students. This increases the range of pedagogical situations to which teachers are exposed during training.

- In order to avoid wasting the teachers' time, tutoring situations that they can easily handle can be made to occur infrequently. For instance, when learning algebra, the simulated student can be rigged to never make an arithmetic mistake or to forget a negative sign.

- A teacher educator can more easily supervise and evaluate novice teachers when the novices are tutoring simulated students, rather than human students.

Most of the activities listed above cannot be done with human students, or even with human actors playing the role of student. Simulated students are necessary for achieving those advantages.

The nature of teacher training itself adds two advantages to the list beyond those normally obtained with simulation-based training.

- The simulations can be rigged to require teachers to articulate, reflect upon and evaluate their implicit theories of learning and teaching. For instance, teachers who used the Buggy game often reported that they used to think that most arithmetic errors were due to carelessness or forgetting, but Buggy convinced them that apparently random errors are often due to deliberate execution of faulty procedures (Brown & Burton, 1978). Teachers who had worked with the Prodigy simulated student volunteered that they "for the first time really understood what some of the research papers from the mathematics and educational psychology courses 'had been on about.' " (Nason, 1991, p. 250).

- Simulations can provide an effective means of facilitating the transfer of educational research findings into classroom practice. The next section illustrates such a transfer, wherein the latest findings on the relationship between explanation and learning are brought to the attention of teachers in a manner that should make the findings easy to understand and clearly relevant to the teachers' practice.

- Teachers can be encouraged to try different teaching strategies with the same simulated student. They might try a larger variety of strategies because they cannot hurt the simulated

student. They can draw stronger conclusions from their experiments because the students' initial knowledge and learning strategies are the same in all trials.

- The simulation can be paused to allow teachers to reflect on their own activity of plan new activities. Moreover, it need not get frustrated or angry. By removing the time pressure and affective stresses that come with tutoring a human student, teachers have more opportunities to become a reflective practitioner and learn from their own activity.

- Certain aspects of the subject matter are always more difficult to learn. By teaching a simulated student, the techer can more rapidly discover the "lay of the land" and in particular, discover mind bugs that are common with the subject matter.

2.3 A speculative example

In order to illustrate some of the advantages of simulation-based tutor training, this section presents a hypothetical scenario of a teacher teaching physics to the Cascade system (VanLehn, Jones & Chi, 1992; VanLehn & Jones, 1993a,b,c; Jones & VanLehn, 1992). Cascade is a algorithmic simulated student that uses a rather fine grain size. Its knowledge of physics is contained in rules much like those in Table 1. It learns these rules by a kind of impasse-driven learning (VanLehn, 1989b, 1991; VanLehn, Jones & Chi, 1992). When it reaches an impasse during problem solving, it uses heuristics to invent a new rule.

Cascade is a fairly mature cognitive simulation. It was originally built to model the self-explanation effect first observed by Chi, Bassok, Lewis, Reimann and Glaser (1989). More recently, it was compared to protocols of 9 subjects as they learned a chapter of physics (Jones & VanLehn, 1992; VanLehn & Jones, 1993a). It is currently being extended to model long-term learning. The goal is to be able to learn about half the material covered in a freshman mechanics course and model many of the psychological findings, including expert-novice differences, the practice effects and the transfer phenomena (VanLehn, 1989a). Although Cascade is currently limited to learning by studying examples and solving problems, we are extending it to learn from a tutor who answers questions and occasionally offers unsolicited advice. Clearly, these extensions are necessary if Cascade is to be used for training teachers. In the rest of this section, we will imagine that these extensions have been completed in order to see how a fine-grained tutor training system might

work.

Suppose the tutor selects the problem shown in the top panel of Figure 1. Cascade starts by drawing a free-body diagram. A free-body diagram should have a vector (arrow) for each force. Cascade first draws the free-body diagram shown in the bottom panel of Figure 1, pausing briefly after each vector in order to allow the tutor to interrupt if she wants to. Cascade assigns coordinate axes. Cascade begins to apply Newton's law by typing $P_y = P \sin 30$ and pausing as usual. This time, the tutor interrupts and selects "No, try again" from a menu because the equation is missing a negative sign. Cascade checks its derivation of the line, and finds that it has no alternative paths, so it had to generate this equation. So it responds, "I can't see what's wrong." The tutor selects "It should be..." from the menu, then edits the equation printed by Cascade, inserting a negative sign to make it $P_y = -P \sin 30$. Cascade responds "ok" but does nothing else.[1]

Place figure 1 about here.

Caption: A physics problem and its free-body diagram

From the brevity of Cascade's response, the tutor suspects that Cascade did not learn anything from the feedback, so she opens Cascade's knowledge browser, which allows her to look through Cascade's rules. Cascade's rules are annotated with their learning history, so the tutor can see when and how each was acquired. The teacher soon determines that Cascade learned nothing from her feedback on the negative sign. Because human students are not equipped with knowledge browsers, the teacher could never be certain what a human tutee learned from her feedback. With the certainty afforded by the knowledge browser, the teacher suspects that this kind of feedback is not always a good tutoring strategy (which it isn't, according to Merrill et al., 1992). She might even realize that simply trying to *tell* Cascade the target knowledge had failed, so she should try to get it to construct its own understanding of the negative sign. (In fact, Cascade is a purely constructivist learner.) Although she could back up the tutoring session and try a different way

[1] Different students have different learning strategies and habits. The current version of Cascade has parameters that control, for instance, how deeply it explains examples to itself (VanLehn & Jones, 1993a). Similar parameters could control how deeply it processes negative feedback, such as that given above. For this illustration, we assume that the parameters are set to make Cascade merely accept negative feedback without trying the understand it deeply or rebut it.

of handling Cascade's error, let's assume that she prefers to continue tutoring and try alternatives after the session is over. The tutor decides to try a risky tactic and selects "Please explain...." from the menu and highlights the line $P_y = -P \sin 30$. This asks Cascade to explain the line. Normally, the tutor only asks students to explain lines that they have written. Here she is asking Cascade to explain a line that she modified in the hope that this will cause it to construct an understanding of the line. She thinks that this tactic probably won't work, so she would not waste precious time on it if she were tutoring a real student. However it is fine to waste Cascade's time, and Cascade will not get frustrated or angry.[2] So the teacher decides to try having Cascade explain a line that it did not entirely produce.

This tactic succeeds. Cascade tries to explain the line, and reaches an impasse trying to explain how the minus sign was obtained. However, the fact that the inexplicable object is a minus sign triggers an overly general rule of mathematics: mathematical operations often preserve signs. The formula $-P \sin 30$ is obtained by projecting the vector P onto the negative part of the y-axis. This is a mathematical operation. The overly general rule suggests that projection onto a negative axis produces a negative term, so Cascade builds the following new rule:

```
If a force vector is projected onto the negative part of the y-axis,
then the resulting algebraic term is negative.
```

This learning event may seem unlikely, but it is based on sound psychological research. Chi et al. (1989) analyzed protocols of students studying examples. They found that when subjects explained solution lines to themselves, they learned much more than they would otherwise. This finding has been replicated and extended many times (Pirolli & Bielaczyc, 1989; Fergusson-Hessler & de Jong, 1990; Bielaczyc & Recker, 1991; Chi, de Leeuw, Chiu & LaVancher, in press; Pressley et. al, 1992). This particular learning event, wherein a rule about projection of negative signs was discovered by explaining a line, occurred with 4 of the 9 Chi et al. subjects (VanLehn & Jones, 1993b) Of the other 5 subjects, 3 knew the rule already and 2 learned it. In short, there is sound psychological evidence for this particular learning event as well as the general mechanism of learning by explaining solution lines to oneself. Needless to say, if teachers are going to learn from a simulation, they must believe that it adequately models human students, so it is important that the psychological research

[2] Although we plan to augment Cascade with a model of motivation, this scenario makes it clear that there are times when one wants to turn the motivation module off.

behind a simulation be sound.[3] The system should probably provide a menu item that explains a learning event to the teacher and cites the appropriate literature. In this fashion, teachers can become aware of important research results in a context that makes it easier to understand both the result and its relevance to their teaching.

This scenario illustrates several things. First, currently available cognitive simulations of learning are adequate for building a simulation-based practice environment for tutors. The simulations are both computationally sufficient for learning and psychologically sound. Second, natural language interaction is not necessary. In this task domain, menu selections, pointing with a mouse, and editing are sufficient. Third, it may not hurt that a simulated student lacks a model of motivation. In fact, it might even encouraged a teacher to try things that she would not try otherwise for fear of frustrating the student. Fourth, not all the technical problems are solved. Cascade, for instance, is too slow for a really natural interaction with human teachers when it runs on a 16Mhz RISC workstation. However, this and similar technical problems are not insurmountable obstacles.

Once such a practice environment is in place, it can be augmented with feedback and hints that will help teachers learn how to tutor. The expert module for such a tutor-training system could even be the tutoring heuristics used by intelligent tutoring systems.

2.4 A tutoring environment based on a large-grained, tabular simulated student

Prodigy is a teacher training system that enables teachers to develop their skill in diagnosing students who have difficulty in fractions arithmetic (Nason, 1991; 1993). The core of Prodigy consists of a set of simulated students. Each simulated student was constructed on the basis of several hours of detailed empirical data from a particular human student who had difficulties with fractions. Each simulated student within Prodigy mimics the behaviors observed in the corresponding human student at a fine grained level. For example, the simulations mimic the handwriting and the writing speed of the observed students. Each simulated student in Prodigy exemplifies a different set of bugs. The computational rules that drive the behavior are implemented in a general and robust way, so that the simulated students behaves sensibly even if it is given fraction problems which were never given to the human students that they are based on.

[3] When teachers played the Buggy game (Brown & Burton, 1978), they often found it hard to believe that such strange subtraction bugs really were found in children's work. Even the developers of commercial diagnostic tests were incredulous. If the game's authors had not had solid empirical work from several large studies, none of these informal trainees would have believed the simulations and hence would have learned little from the Buggy game.

A session with Prodigy begins with the user selecting a simulated student. The user poses problems to that student and observes the resulting behavior. Hypotheses about the simulated student's bugs can be verified by accessing a knowledge browser. However, the Prodigy knowledge browser is based on a different concept than the knowledge brower envisioned for the Cascade system and described in the previous section. In Prodigy, rules are expressed in English that approximates the human student's answers to diagnostic questions. Hence, using Prodigy's knowledge browser is more like listening to the student than it is like opening up the student's head to see what is inside. Prodigy is fully implemented with an easy-to-use interface and runs on an IBM PC. Preliminary trials indicate that student teachers do gain an increased appreciation of the complexity of the relation between observable behavior and the underlying procedural knowledge.

The next version, Prodigy MK2, will let student teachers practice tutoring skills as well as diagnostic skills. This will be accomplished by extending the interface so that the user can input instruction as well as diagnostic problems, and by extending the simulated students with a learning facility that will respond to the instructions. The scenario of working with the extended system will be similar to the hypothetical scenario described for the Cascade system: The student teacher can pose diagnostic problems, construct an hypothesis about the simulated student's difficulties, input some instruction, and then observe the effects of that instruction, either by posing new problems or by inspecting the knowledge directly through the knowledge browser.

Prodigy is interestingly different from Cascade in that it uses a large-grained representation of knowledge consisting of a list of procedural skills, such as

```
the-addition-of-fractions-with-like-denominators
the-generation-of-equivalent-fractions
the-generation-of-lowest-common-denominators
the-conversion-of-fractions-to-new-denominators
```

and with a list of concepts, such as

```
understands-that-fractions-may-represent-equal-sized-parts-of-a-whole
understands-that-denominator-tells-the name-of-the-fraction
understands-that-numerator-tells-number-of-fractional-parts
understands-that-fractions-may-also-represent-sharing-division
```

```
understands-that-fractions-are-numbers-that-show-amounts
understands-notion-of-inverse-proportion
understands-that-same-quantity-is-represented-by-many-fractions
understands-that-tow-quantities-have-many-common-denominator-fractions
```

In Prodigy MK2, the conceptual knowledge determines which procedures it uses. If certain key concepts are missing from the simulated student's repertoire of conceptual knowledge or if the repertoire contains certain misconceptions, then correct procedures may be replaced by incorrect procedures or malrules. Although these incorrect procedures generate incorrect solutions for some problems, they may generate correct solutions for others. Thus, in order to produce a sound diagnosis, the tutor needs to administer and to observe the simulated student doing a wide variety of problem types.

Learning will be represented in a tabular fashion. The learning activities that a tutor can administer to the simulated student are selected from a menu. Upon request, Prodigy will explain what a selected activity contains. For each topic (e.g., comparison of fractions, estimation of addition of fraction, addition of fractions with like denominators, equivalent fractions, etc.), there are four types of lessons, based on Ashlock, Johnson, Wilson and Jones (1980):

- *Initiating activities* provide the student with problems that let him or her improvise solutions using what he or she already knows to solve problems.
- *Abstracting activities* use a range of concrete and pictorial representations of fractions in order to focus each student's attention on their similarities and differences, and to highlight the relationships between the concrete representations and written fraction notation.
- *Schematizing activities* let the students look for and discover interrelationships between the new and previously known concepts and skills.
- *Drill and practice activities* provide practice for concepts and skills.

Prodigy's simulated student does not work through these instructional activities the way Cascade works through physics examples and exercises. Instead, it has a table that indicates what the learning activity's effects on the knowledge representation would be. For example, a drill and practice activity on generating common denominators will only produce short-term increases in

procedural skills because the underlying problem of an inadequate conceptual base has not been addressed by this type of activity. Long term mastery of the processes involved in adding fractions with unlike denominators is achieved only if the teacher also selects and adequately sequences initiating, abstracting and schematizing activities that facilitate the development of the underlying fraction and addition concepts.

Prodigy's table represents a theory of learning. Its cells have been filled in a manner that is consistent with the empirical literature on the learning of fractional arithmetic (e.g., Cuneo, 1988; Hart & Kerslake, 1983; Kerslake, 1986; Post, Wachsmuth, Lesh & Behr, 1985; Ohlsson, 1991, Tatsuoka, 1984). The point here is that when the subject matter is such that a complete algorithmic model of learning is not yet available, one can still produce a viable simulation. Indeed, because the tabular approach is so unconstrained, it might offer even more empirical fidelity than algorithmic simulations, which necessarily use well understood learning mechanisms that are feasible to implement.

3 Collaborative learning and learning by teaching

Simulated students may make it possible to improve upon a traditional teaching method wherein students work together in pairs or small groups. Such collaborative learning or peer learning is quite popular despite the fact that quantitative studies have shown that it produces only modest improvements over traditional classroom teaching. For instance, peer learning produces an improvement of .80 standard deviations over classroom teaching versus 2.0 for human tutors and 1.0 for intelligent tutoring systems (Bloom, 1984; Anderson & Reiser, 1985). Teachers often prefer peer learning because they believe that it improves students in ways that are not easily measured by conventional tests. For instance, peer learning may increase transfer to work places where collaborative work is common, or it may increase students' collaboration skills. Regardless of the reasons for its popularity, the quantitative studies show that there is room for improvement in peer learning along cognitive dimensions.

3.1 What advantages would a simulated peer have over a human one?

The technology exists to substitute simulated students for one or more of the students in a collaborative learning situation. But why should we bother? What are the cognitive dimensions along

which peer learning can be improved? Can simulated students move the peer learning situation along those dimensions?

One common finding in studies of collaborative learning is that the students who teach more also learn more. For instance, Webb (1989) found that learning correlated more with the number of times a student *answered* a question of another student than with the number of times a student *asked* a question (for similar findings, see O'Donnell et al., 1990; Pressley et al., 1992). This result is counterintuitive. One would expect the people seeking knowledge (question askers) to gain more knowledge than those dispensing knowledge (question answerers). The finding that question answerers learned more is reminiscent of the Chi et al. (1989) finding that students who explained physics examples learned more effectively. Apparently, explaining things both to oneself or to another student helps one's understanding.

This finding about collaborative learning implies that one way to improve it is to put more students into the question answerer role. Suppose one of the students in a pair learning situation is a simulated student. It can ask questions of the human student. Answering the questions should increase the human student's learning. Of course the simulated student should avoid pestering the student with questions about knowledge that the student has already mastered. It should also avoid asking about topics that are totally unfamiliar to the human student. A really sophisticated simulated student would maintain a model of the human student and only ask questions that are pedagogically appropriate given what it believes the student knows. That is, it would maintain a traditional student model. However, it may be difficult to obtain enough information about the human to maintain a student model. An equally effective tactic might be to use the simulated student's knowledge, which evolves as the simulated student learns, in place of a traditional student model. Suppose that the human and simulated student had been collaborating for a long time. They will have studied the same material, solved the same problems and even produced the same solutions. They should have approximately the same knowledge.[4] If the simulated student only asks questions about topics that it is genuinely uncertain about, then those may be just the right

[4]This technique might work for regular intelligent tutoring systems. The current practice in student modeling is to use primarily the student's actions to fit the student model, and either ignore the instructional activities entirely or to use them merely to bias the interpretation of student actions. If a simulated student for the instruction exists, then one could reverse the priorities. For instance, if the simulated student has just learned something, then the student modeler could assume that the human student just learned the same thing, until the human student's actions prove otherwise.

questions to ask the human student in order to increase that student's learning.

The discussion above has focused on just one kind of interaction: question asking and answering. Research in small work groups has found many kinds of interactions that are correlated with learning rate (see Pressley et al., 1992, for a review). Simulated students can probably be designed to increase the frequency of many beneficial interaction types.

Simulated students provide an antidote to a serious problem with practical uses of peer group learning. Once a teacher has placed students in a group and given them a task, there is little control over the group's interactions because the teacher can only spend a fraction of his or her time with that group. With a simulated student as part of the group, all kinds of pedagogically beneficial interactions can be staged from within the group itself – thought provoking questions can be asked, taciturn students can be prodded to speak, bad ideas can be questioned, small slips can be caught before they have serious consequences, attention can be directed away from areas that are already mastered and onto areas where students are ripe to learn, and so on.

Most of this control can be exercised because a simulated student can have one thing that a real student can never have: a second knowledge base that is nearly complete and correct. When deciding how to act, it refers both to its "own" knowledge base and this expert knowledge base. The expert knowledge base is used, for instance, to determine whether the current line of reasoning is either correct, incorrect but harmless, or incorrect and destined to lead to serious confusions. This expertise is necessary for the teacher when he or she guides a small group, so it should be necessary for the simulated student as well. The lack of such expertise in a group composed only of human students dooms it to be less effective than one with a simulated student, in principle at least.

3.2 Limitations

Naturally there are limitations on the participation of simulated students in peer learning that are due to the current state of technology. As discussed earlier, the state of the art in natural language processing, and speech in particular, is not sufficient to allow machines to participate in the kinds of open-ended discussions that groups are famous for. This means that machines will only be able to participate in discussions that are mostly formal. As a litmus test, if one can imagine a small group of human students successfully working on a problem when none of them speak the same

language, then one has found a task domain where simulated students could participate as equals. (An example of such a task domain will be mentioned in a moment.) It is important to allow the simulated student and the human student to "see" the same screen and "point" to it.

Another limitation is that large-grained simulated students, which are easier to build than small grained-ones, will probably not work as participants in a small group. For instance, if the simulated student asks a human student a question about fraction addition, and the human student selects the menu item, "use manipulatives to review meaning of common denominators," the human student has in some sense answered the machine's question, but the human student has probably not learned as much as would be learned if the human actually carried out the demonstration with manipulatives. To support a natural dialogue at a pedagogically effective grain size, the simulated student would probably have to use a fine-grained knowledge representation.

A tabular simulated student can be used as a peer learning partner, although it may not be as accurate a model of learning as a algorithmic model. Chan and Baskin (1990) suggest modeling learning by advancing the simulated student along a prefabricated sequence of knowledge bases. Advancement could be made in response to the simulated student's experience or in order to keep pace with the human student's learning as monitored by some kind of student modeling. A more incremental approach is to attach to each rule in an expert model a switch or weight that controls whether the rule is active or not. Initially, most of the simulated student's rules would be inactive. As in Prodigy MK2, learning is simulated by switching on rules or increasing their weight until it exceeds a threshold. Both kinds of tabular simulated student can be implemented without machine learning technology, which makes them easier to develop. However, they would probably not be as accurate in portraying a student's learning as a algorithmic approach, which is why they would not be advisable for fine-grained teacher training. On the other hand, accurate modeling of learning may not be important for peer learning applications if all one requires is that the simulated student's competence keep pace with the human student's competence as the human student learns. If one wants the simulated student to demonstrate good learning strategies in the hope that the human student will begin to use them as well, then of course a algorithmic model of learning will be necessary.

3.3 A simulated physics student collaborating with a human

As an illustration of the benefits that could be derived from a simulated peer learner and the challenges involved in creating one, this section describes a hypothetical interaction between Cascade and a human student. The illustration shows that a natural interaction can be held with only limited natural language processing.

Suppose that the human student is looking at the interface shown in Figure 2. This interface is actually just the problem solving interface used by the Olae student modeling system (Martin & VanLehn, 1993), with an extra window at the lower left for dialogue. The top bar holds icons for problems. The upper right window shows the selected problem. The lower right window is for drawing free-body diagrams. The upper left window is for entering equations. The lower left window records the dialogue between the simulated and human students.

After exchanging greetings, Cascade prints "How about working on this one?" in the dialogue window and selects one of the problem icons at the top of the screen. The selected problem is displayed and the free-body diagram is initialized (as shown in Figure 2). Although the human student could disagree by selecting another icon, let's suppose that she doesn't.

Place figure 2 about here

Caption:The screen after Cascade has suggested a problem to solve

---

Suppose both Cascade and the student know that the first step in solving force problems is to draw the forces onto the free-body diagram, but they have to negotiate who will do it. Cascade pauses a moment to see if the human will take the lead, and when the student types nothing, it prints "Shall I draw the forces?" This somewhat stilted language is used over the more colloquial "Do you want me to draw the arrows?" because it is shorter and thus faster to read, and it is less ambiguous. The goal is to expedite communication, not to simulate a natural language conversation. The student types "yes" and Cascade draws the forces shown in Figure 3, pausing after each to allow the student to interrupt if she wants.

Place figure 3 about here.

Caption: The free body diagram, with the normal force missing

---

When Cascade has finished drawing the forces, it covertly checks the correct solution, which has been derived earlier using an expert knowledge base. Cascade observes that a force is missing from the free-body diagram that it drew. Cascade, and presumably the human student as well, had earlier acquired a rule for drawing normal forces.[5] However, Cascade has never used that rule to draw a normal force, so the rule's strength in memory is weak and it did not fire. Cascade reasons that the human student may also have a weak rule for normal forces, so it decides not to move on to equation generation, which is the next step in the procedure for solving force problems. Instead, it asks the human student for confirmation of its actions by printing "Okay?" in the dialogue window. The human selects the menu item, "No, it should be..." and draws the missing force on the free body diagram. This interaction strengthens Cascade's normal force rule and presumably the human student's rule as well.

However, unlike a human collaborator, Cascade plays dumb and asks the human, "What's that?" The pedagogical purpose here is to get the human student to explain the force-drawing action, which is a good way to get students in peer learning situations to increase their learning rate. In this case, for instance, the human student might have drawn the force by making some kind of visual analogy with her memory of the example's free-body diagram. In answering Cascade's question, she should recall why that force was drawn on the example's free-body diagram. Thus, explaining the force instead of just drawing it should strengthen the student's knowledge about normal forces.

The human student pauses a moment then types, "the normal force." Understanding explanations like this one requires that Cascade do some natural language processing. However, the context is so specific that such processing may be feasible.

Because the student's explanation agrees with the expert's, Cascade accepts it, and prints "ok." Now it moves to the next step of the force procedure, which means negotiating who will do it. Having no particular pedagogical motives at this point, Cascade uses the heuristic that the two of

[5] When a body is supported by a surface, there is a force exerted on the body by the surface. That is, the surface pushes on the body in order to counteract the gravitational force and keep the body from falling. This force is called the "normal" force because it is perpendicular (normal) to the surface.

them should share the work evenly, which suggests nudging the human student to take responsibility for the next step. Cascade types, "Your turn?" This way of turning over control of the dialogue is quite clumsy compared to the subtle pauses, noises, facial expressions and body movements used by humans in face-to-face collaboration (Merrill et al., 1992). However, it is probably the best that can be done with current user interface technology. At least it is fast and unambiguous.

The human student types $P_y = P \sin 30$ in the solution window (the upper left window of Figure 2). Cascade checks the expert's solution and finds that there should be a negative sign before the P. Its rule for the sign is quite strong, so it suspects that the human student has made an unintentional error. It types "oops" and waits, expecting the human student to correct the error. After five seconds, the human hasn't done anything, so Cascade infers that the error hasn't been detected. It politely prints "Negative sign?" expecting that this would jog the human student's memory and prompt her to edit the line. Again, nothing happens, so Cascade types "I think it's ..." and edits the line itself. But when it asks, "Okay?" the student disagrees. Cascade and the human student now have a genuine disagreement. Cascade considers what would happen if it backed down and let the sign error persist. It looks ahead using both its rules and the expert's, and finds that the error would affect the answer but not the solution path. Cascade decides that letting the sign error persist will not have pedagogically disastrous effects, so it prints, "Okay, we'll do it your way. But let's ask the teacher about it later." Cascade sends electronic mail to the teacher describing the problem and requesting help. When the teacher arrives sometime later, Cascade can pull up the screen at the time of the sign error, explain the disagreement, and let the teacher take over. Presumably, the teacher will remediate the human student's misconception.

This scenario shows that managing a peer problem solving session requires much more than just a simulated student. Indeed, the learning capabilities of the simulated student play only a small role in the hypothetical system (except for maintaining a model of the human student). Much of the system's reasoning concerns management of the dialogue. However, the scenario also illustrates that in a formal task domain, such as physics, a pedagogically useful dialogue can take place without much natural language processing. Indeed, there are many times when the use of a text medium for a conversation implies using more concise language and less subtle cues than are used in real human-to-human conversations. This suggests that it is probably not necessary to fully understand how human-to-human dialogue works in order to build a peer problem solving system.

3.4 Simulated peer learners: The larger picture

Although the idea of a simulated learning partner is not new (Chan & Baskin, 1990; Gilmore & Self, 1988; Cumming & Self, 1991), few implementations exist. Dillenbourg and Self's (1992b) system teaches a student how to rearrange electoral districts to maximize the number of seats held by a political party. The human student and the simulated student are working for the same party, and they argue over which redistricting tactics are most likely to increase their party's winnings. Hintze' (1991) system is configured more like a help system than a collaborating partner, even though Hintze calls it a co-learner (Dillenbourg's term for a simulated learning partner). The human student tries to discover the function of the keys on a fancy telephone. The system checks the student's hypotheses and sometimes suggests one of its own. Palthepu, Greer and McCalla's (1991) system acts like an interactive knowledge acquisition tool for inheritance hierarchies. The student tells the system facts, such as "mammals have legs," "humans are mammals" and "dolphins are mammals." The system asks questions in order to complete the inheritance hierarchy (e.g., "Are humans and dolphins the only mammals?") and checks the hierarchy for consistency (e.g., "Do dolphins have legs?"). All three systems are similar in that they contain only two components: a learning module that updates the simulated student's knowledge base, and a dialogue control module that decides what to say next. None of the systems contain an expert module, so none know when the students have adopted a misconception or are wasting time exploring unproductive territory. In a sense, these are *pure* co-learners. As illustrated by the scenario above, a more complex system could use covert expert and pedagogical modules to move the dialogue in pedagogically useful directions.

It is tempting to conclude that pure co-learners, which have access to the same information as a human peer, could not be any better than humans at expediting their partners' learning. On this line of reasoning, a pure co-learner could at best offer only .80 standard deviation improvement over classroom teaching, which is not as good as the 1.0 improvement of intelligent tutoring systems. This conclusion would be unwarranted. The .80 improvement is an average over all students in the peer learning situation. However, some students learn more than others (Webb, 1989). The earlier discussion suggested the better students learn more because they are *answering* more questions. There are probably other behaviors that also differentiate effective learners from ineffective ones. If a pure co-learner can increase the amount of time human students engages in question-answering

and other productive behaviors, then they will learn more with a simulated partner than with a human partner. The Palthepu, Greer and McCalla (1991) co-learner, for instance, is explicitly designed to ask many questions of the students in the hope that some of them will challenge the student to think more deeply about their beliefs, elaborate them and come to a better understanding of the task domain.

The more complex co-learner envisioned earlier requires as much engineering as an intelligent tutoring system does. It requires a good user interface, an expert module and a pedagogical module, but it uses a simulated student in place of a student module. There may be a very fine line between such co-learners and really good intelligent tutoring systems. Studies of expert human tutors often show that they act more like a peer in a collaborative learning situation than a pedantic dispenser of feedback and explanations (see Merrill et al., 1992, for a review). This opens the exciting possibility of a highly adaptive system that acts like a peer with some students and like a tutor with others. If it contains the capabilities to be both, then it is only a matter of adjusting a dialogue parameter. Indeed, students could try working with several simulated partners, and choose the one they like best. The set of potential partners could vary both in how passive they are and in how much expertise they display.

As Chan and Baskin (1990) point out, one could have a simulated learning group, with multiple students and teachers. They present a three-agent learning situation: a human student works with both a simulated student and a simulated tutor. Such systems, which they call learning companion systems, require the usual 4 modules of an intelligent tutoring system, plus an additional module for the simulated student. Chan and Baskin describe a particular learning companion system that teaches integral calculus. It is not clear how its simulated student is implemented, as the authors discuss both a algorithmic and a tabular approach. The authors' focus is on the interactions among the three agents. Chan and Baskin envision a variety of patterns of interaction. Usually the human student and the simulated student work together on problems and ask the teacher for help only when they get stuck. The teacher chooses problems and critiques the students' solutions, but rarely interrupts the students' problem solving. However, when problems are easy, the two students each produce their own solutions, then check each other's work. The teacher then checks the solutions in order to correct any errors that the students overlooked.

So far we have argued that simulated students/tutors allow one to construct learning groups

that will have productive interactions, assuming that we know from laboratory studies which kinds of interactions are productive. However it is also possible to use such controlled groups to conduct the laboratory studies that will tell us which kinds of interactions are productive. In the long run, this might help both human-only peer learning as well as simulation-based peer learning.

4 Simulation-based formative evaluation

Instructional design is the process of creating courses, textbooks, tutorials, tutoring systems and other types of instruction. Textbooks on instructional design (e.g., Dick & Carey, 1985) emphasize that designs should be pilot tested as early as possible. Such "formative evaluations," as they are called, should uncover defects early in the design process when they can still be corrected.

Unfortunately, formative evaluations are seldom done (Komoski, 1974), and for at least three good reasons. Most instructional design involves a great deal of planning before the actual production of the artifact. For instance, one outlines a course before assembling the readings and generating the homework exercises. One would like to have feedback on the major design decisions long before the actual production of the instruction, but when formative evaluations are conducted with human students, they can only be conducted with the actual instructional artifact. You cannot give an outline of a textbook to pilot subjects and expect to get reliable information on its merits and defects. In short, formative evaluations can only be conducted after some production has taken place, and by then it may be too late to revise major design decisions.

The second reason that formative evaluation is seldom done is that results from the pilot subjects are not as informative as they need to be (Twidale, 1993). A thorough evaluation administers pretests and post-tests to the students, but this only allows comparison of the new instruction with older instruction. Moreover, the comparison is based on the assumption that the tests measure what the instruction is designed to teach. One can simply ask the students for their suggestions, but students often do not even know that they are confused and hardly ever know what confused them. Thus, only the most obvious defects in the instruction are uncovered by interviewing students, and those defects may not be the most important ones. Formative evaluation should be a form of troubleshooting. It should tell the development team exactly what is wrong and why. That information is just not available from test scores or student interviews.

Lastly, formative evaluations are rare because they are expensive. Much instruction, particularly

in industry and government, is turned out on tight budgets and schedules. The organization cannot afford the several months that are required to round up pilot subjects and run them through the instruction.

Simulated students may remedy all these problems that thwart formative evaluation. They should provide cheap, fast formative evaluations that pinpoint defects in the instruction and can work on designs that are only in the planning stages. The idea is simply to have simulated students "take" the instruction and report on what they learned. Moreover, a formative evaluation could use multiple simulated students so that the designer can understand the interaction of the instruction with differing amounts of prior knowledge and different learning strategies (i.e., understand aptitude-treatment interactions).

Ideally, an instructional designer would have the same tools that an engineer has. An engineer's CAD station can be networked with a finite element analysis program, a simulated wind tunnel, or some other simulation-based analysis program. When the engineer is satisfied with a design, she pushes a button and sends it off for analysis. Some time later (it can be days), the results are available in the form of reports and a database that the engineer can interrogate in order to find out how the design performed. Such simulation-based testing is not a substitute for building the real machine and testing it because no simulation is perfect. However, the pinpoint-analysis and the ability to evaluate designs early in the design process outweigh the fidelity issues, making formative evaluation during engineering extremely cost effective. Nowadays, the instructional designer can get the equivalent of CAD stations with commercially available authoring systems or better still, design systems that support planning as well as production of instruction (see Pirolli, 1991, for a review). However, none have a button that one can push in order to obtain a formative evaluation. The technology for simulation-based formative evaluation is here, but the systems themselves still need to be built. They do not have to have perfect fidelity, but they should offer precise analyses of instructional defects and they should work with early designs.

4.1 Limitations

The current technology imposes some major limitations on simulation-based formative evaluation. As in simulation-based teacher training and simulation-based peer learning, the computer's limited natural language processing capabilities restrict the types of formative evaluation that can be done.

However, this might not be a big problem. As argued earlier, formative evaluation is best done with outlines or early designs, rather than the actual material used by human students, because designers need early feedback on their designs. This means that the instruction presented to the simulated student need not be expressed in natural language. It could be expressed in a restricted or formal language instead.

A second and more serious limitation is due to the current state of the art in machine learning. Learning of substantial knowledge by machines is a fairly recent advance, and most systems employ restricted knowledge representations or algorithms that limit the kinds of knowledge they can acquire. For instance, Cascade (VanLehn, Jones & Chi, 1992; VanLehn & Jones, 1993a,b,c; Jones & VanLehn, 1992) can learn classical Newtonian mechanics. It could probably learn closely related topics, such as thermodynamics or aerodynamics, with only small changes. At one time, it learned a little bit of combinatorics and elementary probability theory. However, there are many topics that it could not learn without a complete overhaul.

The grain size of the simulated students determines the grain size of the instructional designs they can test. If the simulation represents knowledge in terms of fine-grained rules, then the designer will probably have to work the design out all the way to sentence-sized assertions, examples and exercises. If the simulation represents knowledge in terms of coarse-grained sets of skills or concepts, then the instruction can be represented in a coarse grained way. For instance, the designer might specify the number of exercises and the skills used, rather than generate the actual exercises. Ideally, a formative evaluation system would have simulated students at several granularities so that it could test designs at several points in the design process.

Algorithmic simulations of learning, rather than tabular ones, are probably necessary for formative evaluation. Earlier we noted that tabular simulations should work well for training teachers because teachers' interactions with the system could be constrained so that every unit of instruction that they submit occurs somewhere in the system's tables and thus allows the system to produce appropriate changes in its knowledge state and behavior. Teachers can experiment with different sequences of instructional units, but they cannot devise and submit a new instructional unit. This limitation would probably be too strong for instructional designers. Their business often entails creating novel instructional units, and these units would not occur in the tables of a tabular simulated student. Most formative evaluation applications require algorithmic simulations.

The result of these technical limitations is that a simulation-based formative evaluation will probably have to be limited to a single class of task domains, such as mathematical analyses (Cascade's natural class), text editing procedures, electronic troubleshooting, familiarization with equipment, etc. Moreover, the simulation will probably have to be algorithmic, avoid natural language, and have the right grain-size for the kinds of instruction designs that need to be tested.

4.2 An illustration of simulation-based formative evaluation

Sierra was a machine learning program designed to model young children's learning of arithmetic, elementary algebra equation solving and other simple mathematical calculational skills (VanLehn, 1987, 1989b). Sierra was a fine-grained algorithmic model rather than a tabular model. It learned correct procedures as well as buggy ones, just as children do.

Sierra learned from solved examples. For instance, the subtraction lessons of three major textbooks were each given to Sierra. All three textbooks introduced borrowing in problems that had only two columns, such as

```
  8 5
- 2 7
------
```

This gave Sierra trouble. It could see that the digit borrowed from was always in the upper left corner (the 8, in this case), but it did not know how to generalize that location. Was it supposed to be the top digit in the *left-most* column or the top digit in the column *adjacent* to the column initiating the borrowing? Sierra ignored the explanations that accompany borrowing because they required an understanding of the base-ten system, which Sierra (and many young children) lacked.[6] When Sierra was given 3-column problems that require borrowing in the units column, such as

```
  7 4 5
- 1 2 7
---------
```

[6]Even when the base-ten system is taught carefully, some students ignore the semantics of arithmetic procedures, focus on syntax, and acquire bugs (e.g., Resnick & Omanson, 1987). Sierra models the syntactic learning of such students, so its suggestions for improving instruction will tend to block bug formation by students who approach arithmetic learning syntactically. Its suggestions can easily be incorporated into a semantically-based curriculum and should create a "safety net" to catch such syntactical learners.

it would not know whether to decrement the tens column (because it was adjacent to the column causing the borrowing) or the hundreds column (because it was the left-most column). This confusion causes Sierra to get stuck (reach an impasse). Students and Sierra employ a variety of ways to work around the impasse, including omitting the borrow-from action, omitting the whole borrow, or just picking one column and hoping that it is the right one for the borrow-from action. The different repairs to the impasse correspond to different bugs. In one large study, 4 of the predicted bugs were observed, and two of them were moderately common, occurring in the work of 6 students each. Thus, Sierra was able to explain how several observed bugs were acquired.

Such explanations of bug origins can be viewed as evaluations of the three commercial textbooks. They indicate exactly where some misconceptions were learned and why. In particular, the case just given suggests that textbooks should introduce borrowing in three column problems, such as $745 - 127$. This would disambiguate where the borrow-from should be placed. Indeed, when Sierra was taught borrowing with such a lesson sequence, the 4 observed bugs were no longer generated. Thus, Sierra can not only troubleshoot instruction, it can also indicate when a modification to the instruction fixes the problem. This is exactly the kind of advice that an instructional designer needs.

It is doubtful that such precise advice could be obtained from formative evaluations with human students. Few students with a bug could tell an interviewer how they learned it. In order to uncover instructional defects through test scores, a very large number of students would have to be tested and a very detailed analysis would have to be conducted. Worse yet, many bugs are intermittent. Students make different repairs to the same impasse at different times. Sometimes further instruction causes students to abandon a bug, although it may resurface later. Thus, if testing is not done throughout the instructional period, some defects in the instruction will be nearly impossible to detect. Testing at the end of the instructional period will portray students as having a large variety of errors that appear and disappear for no apparent reason. Longitudinal testing throughout the instruction disrupts the instruction, and cross-sectional testing requires very large numbers of subjects. So it is nearly impossible to get enough test data to uncover intermittent bugs. For many reasons, then, human-based formative evaluation cannot yield the kinds of precise advice that instructional designers need.

A real arithmetic textbook lesson has perhaps 10 examples and 25 exercises. The text contains

graphics and other devices to motivate the students. The teacher may boost class involvement by having students work problems on the board and by using real-world problems. Considerable planning and material must be prepared in order to conduct a lesson with human students, even if the lesson is only a formative evaluation. However, Sierra can evaluate a lesson while it is still in the planning stages, long before all the material is generated and lesson plans made. Because Sierra models neither practice effects nor motivation, it can learn from a lesson that has only, say, 4 examples and 6 exercises. It takes a designer only a few minutes to draft a lesson using Sierra's editor. A whole lesson sequence can be created in an hour or two. The point here is that simulation-based evaluations can be conducted when instruction is still in draft form. If formative evaluations must wait for all the drill, graphics and other supporting material to be generated, it may be too late to make any difference in the design of the ultimate product.

4.3 A broader view of simulation-based formative evaluation

The Sierra discussion illustrated two of the advantages of simulation-based formative evaluations over human-based formative evaluations: They yield more precise advice and they can be conducted earlier in the design process. In order to illustrate a third advantage, we turn to another system, HS (Ohlsson, Ernst & Rees, 1992). It shows how simulation can let the designer analyze the interaction of the proposed instruction with prior knowledge or other individual differences.

Ohlsson et al. simulated the interaction of prior knowledge of the semantics of subtraction with two kinds of instruction. That is, they took four simulated students, two that understood the meaning of subtraction and two that could only understand subtraction as symbol manipulation. For each pair, they taught one member of the pair the traditional decomposition algorithm, in which borrowing-from is accomplished by decrementing the top digit in the adjacent column. They taught the other member of the pair the equal-additions algorithm, in which borrowing-from is accomplished by incrementing the bottom digit in the adjacent column. The decomposition algorithm is more widely taught, in part because Brownell (1947) and Brownell and Moser (1949) argued that the decomposition algorithm, while harder to execute, should be easier to learn because it was based more directly on the semantics of the base-ten system. Ohlsson et al. found the opposite result. They found that the equal additions method was always easier to learn, and particularly easy to learn when the simulated student understood the semantics of the base-ten

system. In addition to questioning a widely held belief in the mathematics community, the HS work shows how simulations can reveal interactions of instruction with prior knowledge. This is valuable information for an instructional designer to have.

The different systems described so far take different kinds of instructional input. Sierra processed instruction consisting mostly of examples and exercises. This illustrates the limitation mentioned above, which is that instruction that uses formal rather than natural languages is easier to work with. HS (Ohlsson, Ernst & Rees, 1992) models tutorial feedback by assuming that it is the content of the feedback messages rather than their precise form that matters. It encodes the messages as formal constraints, which specify what should be true of the current problem solving state and why the constraint is relevant.

Even natural language instruction can be evaluated with simulations. Kieras (1992, 1989) has developed a system that reads natural language text from technical manuals, such as the "owner's manual" for the T-38 airplane. It evaluates the text for its comprehensibility, which is measured by counting the number of nominal references that can be successful resolved given only syntactic information and a small amount of general knowledge (e.g., that airplanes have wings). More importantly, it points out specific problems with the text and suggests ways that the instructional developer can improve it. Although the Kieras system does not actually learn anything from the text it reads, Mayer and Kieras developed a system that learns how a specific electronic circuit works by reading text[7] and a schematic diagram about the circuit (Mayer, 1990). Learning is facilitated if the system can use knowledge acquired earlier. This allows the system to be used for formative evaluation of instructional sequences. Kieras (1991) compared the predictions from Mayer's model with data from human subjects. The results were encouraging, but larger-scale experimentation is needed. The Kieras and Mayer work shows that even natural language instruction may be amenable to evaluation with simulated students in the very near future.

All the learning systems discussed here work only with a narrow range of subject matter. Sierra can only learn procedures that manipulate written, formal symbols. HS is more general, but still restricted to learning simple procedural skills. This limitation reduces the utility of these systems for formative evaluations. However, it is much easier to generalize an existing system than to build

[7] Actually, the system did not read natural language text, but instead was given the kind of propositional encoding that would be produced by the Kieras system if it read the text.

one from scratch, so with only modest effort, the generality of any of these systems could probably be significantly increased.

Nonetheless, there will always be some limitations on the kinds of material that can be learned by a simulated student. This implies that simulation-based formative evaluation may be most practical in two cases:

- If designers produce many pieces of instruction in a single area, they can be evaluated with the same simulated student. For instance, a simulated student that is capable of learning about the operation of electronic circuits can be used to evaluate instructional material for many different pieces of equipment.

- If the instruction must be of extremely high quality, it may be worth building a simulated-student just for it. For instance, so many students must learn how to sound out words, that it might be worthwhile building a simulated-student that learns pronunciation rules and exceptions, then use it to evaluate proposed sequences of readings.

5 Conclusions

Only a modest amount of experience with applications of simulated student exists. However, what there is warrants the following conclusions.

Given a machine learning system that learns in a sufficiently human fashion, the simplest applications to develop should be tutor training systems. A basic trainer would simply allow the tutor to practice teaching the simulated student. A slightly more elaborate system would allow the tutor to view the simulated student's knowledge, reset the student's knowledge to an earlier state, and replay stored tutorial sessions. Such a practice environment would set the stage for an intelligent tutoring system that coaches teachers to become better tutors.

Simulated students can be used by human students as a learning partner or co-learner. The machine could work collaboratively with the human, covertly manipulating the activity to enhance the student's learning (e.g., by asking its human partner questions at just the right times). This application requires that the machine learning system be augmented with some kind of dialogue module that decides what the machine should say next. The dialogue module should be designed not to generate naturalistic conversations, but to expedite the human student's learning. It may

take considerable work with pilot subjects in order to achieve this goal. For instance, having the simulated student ask lots of questions should promote learning, according to psychological studies (Webb, 1989; Chi, de Leeuw, Chiu & LaVancher, in press), but asking too many questions might irritate the human student. Thus, this particular application of machine learning probably requires more additional work than the application of simulated students to teacher training or formative evaluation. On the other hand, because there are many more students than teachers or instructional designers, the higher development cost may be offset by wider benefits.

The second most simple application of an existing machine learning system would be formative evaluation. Instructional designers submit their instructional prototypes to the simulated student, and it finds places where it cannot learn or learns the wrong thing. In order to allow a useful range of instruction to be evaluated, the machine learning system should be able to handle a variety of subject matter and a variety of instructional types. Given the narrow competence of many machine learning system, obtaining this generality might require additional programming.

All three applications can be based on fine-grained simulations, wherein knowledge is represented in small pieces (e.g., "if a taut string is tied to a body, there is a tension force on the body"). However, it may be much easier to build a coarse-grained simulated student that represents knowledge in larger pieces (e.g., has-mastered-kinematics, understands-reference-frames). Large-grained simulations may work for teacher training and some formative evaluation, but probably not for co-learning systems.

All three applications can be based on algorithmic simulations, wherein a theory of learning is embedded in procedures for learning. However, theories of learning may not be readily available in some application areas. For these, it may be possible to use a tabular simulation, where the effects of specific instructional treatments on specific pieces of knowledge are recorded in tables. Tabular simulations can be used for teacher training and for co-learning systems with modest ambitions, but probably not for formative evaluation.

A major consideration in applying simulated students is their limited ability to understand natural language. This constrains the choice of subject matter and instructional mode. We can model learning of mathematical problem solving from examples and exercises, but we cannot model learning ethics from group discussions. This constraint affects all three applications equally.

A second limitation with current models of students is the absence of any models of motivation.

Current machine learners do not get bored or excited. This limitation may be soon removed as AI begins to model affect and emotions (e.g., Ortony, Clore & Collins, 1985). Meanwhile, the lack of motivation models places strong constraints on applications of simulated students in teacher training and perhaps formative evaluation. It is probably less of an issue in co-learning, where the simulated learner's reaction should be determined by pedagogical considerations rather than its own personal interest in the material (if it had any).

Lastly, all the applications are affected by the current lack of computational models of learning in a group. Our belief is that the learning processes that are observed operating when a single student is working alone with examples, exercises or text, also operate when a person is learning in a group that is arguing, collaborating or discussing those same examples, exercises and texts. There are of course obvious differences in the two situations, but whether those differences really matter to the learning outcomes is not yet known. On the other hand, it could be that learning is radically different when it occurs in a group. It is an empirical question that may take decades to answer. Current work on multi-agent learning in machine learning will provide the foundations for a simulation-based exploration of the issue. Meanwhile, it is safest to restrict applications of simulated students to situations where a single student is working alone with passive instructional material or with a single agent, either a tutor or a co-learner.

We have focused on three applications of simulated students, but there are others that we haven't the space to discuss fully. For instance, Kieras (1990) shows how simulated students can be used to obtain specifications of the knowledge required to do a task, or to evaluate the adequacy of proposed specifications of the body of knowledge. The Cascade simulated student is being used in the Olae student modeling system (Martin & VanLehn, 1993). Our main point is that cognitive simulation has come of age. Although there are still limitations that will take significant basic research to remove, simulation technology can and should be used to solve educational problems now.

6 Acknowledgements

The research reported herein was supported by the Cognitive Science Division of the Office of Naval Research under grant number N00014-91-J-1552 to the first author and grant number N00014-89-J-1681 to the second author. The research of the third author is supported by grant A79130255

from the Australian Research Council. We appreciate the comments of Pierre Dillenbourg, Joel Martin, Jon Rubin and Sigalit Ur on drafts of the manuscript.

7 References

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

Anderson, J.R. (1988). The expert module. In M.C. Polson & J.J. Richardson (Eds.) *Foundations of intelligent tutoring systems.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J.R., Greeno, J.G., Kline, P.J. & Neves, D.M. (1981). Acquisition of problem-solving skills. In J.R. Anderson (Ed.) *Cognitive skills and their acquisition.* pp. 191-230. Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J.R. & Reiser, B.J. (1985). The Lisp tutor. *Byte*, April, 159-175.

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

Ashlock, R.B., Johnson, M. L., Wilson, J. W. & Jones, W. L. (1980). *Guiding each child's learning of mathematics: a diagnostic approach to instruction.* Columbus, OH: Charles E. Merrill.

Badre, N.A. (1972) *Computer learning from text.* Technical Report. Berkeley, CA: University of California, Electronics Research Laboratory.

Behr, M.J. & Post, T. R. (1988). Teaching rational number and decimal concepts. In T. R. Post (Ed.) *Teaching mathematics in grades K-8.* Allyn and Bacon, Boston, MA.

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,* Charlottesville, VA: Association for the Advancement of Computing in Education.

Bloom, B.S. (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.

Brown, J.S. & Burton, R.R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science, 2*, 155-192.

Brownell, W.A. (1947). An experiment on "borrowing" in third-grade arithmetic. *Journal of Educational Research, 16*, 161-263.

Brownell, W.A. & Moser, H.E. (1949). *Meaningful vs. mechanical learning: A study in Grade III subtraction.* Durham, NC: Duke University Press.

Carlson, R.A., Sullivan, M.A. & Schneider, W. (1989) Practice and working memory effects in building a procedural skill. *Journal of experimental psychology: Learning Memory and Cognition, 15*, 3, 517-526.

Chan, T.W. & Baskin, A.B. (1990). Learning companion systems. In C. Frasson & G. Gauthier (Eds.) Intelligent tutoring systems: At the crossroads of artificial intelligence and education. pp. 6-33. Ablex: Norwood, NJ.

Chi, M.T.H., Bassok, M., Lewis, M., Reimann, P. & Glaser, R. (1989) Learning problem solving skills from studying examples. *Cognitive Science, 13*, 145-182.

Chi, M.T.H., de Leeuw, N., Chiu, M., & LaVancher, C. (in press). Eliciting self-explanations improve learning. *Cognitive Science.*

Cumming, G. & Self, J. (1991). Learning modeling in collaborative intelligent educational systems. In P. Goodyear (Ed.) *Teaching knowledge and intelligent tutoring.* Norwood, NJ: Ablex.

Cuneo, D.G. (1988). *Understanding fraction addition.* Paper presented at the Annual Meeting of the American Educational Research Association. New Orleans, LA.

Dick, W. & Carey, L. (1985). *The systematic design of instruction.* Glenview, IL: Scott, Foresman and Company.

de Corte, E., Verschaffel, L., & Schrooten, H. (1991). Computer simulation as a tool in studying teachers' cognitive activities during error diagnosis in arithmetic. In P. Goodyear (Ed.) *Teaching knowledge and intelligent tutoring.* Norwood, NJ: Ablex.

Dillenbourg, P. & Self, J. A. (1992a). A framework for learner modeling. *Interactive Learning Environments, 2*, 2, 111-137.

Dillenbourg, P. & Self, J. A. (1992b). A computational approach to socially distributed cognition. *European Journal of Psychology of Education, 7*, 4, pp. 353-372.

Doak, E.D., & Keith, M. (1986). Simulation in teacher education: The knowledge base and the process. *Tennessee Education, 16*, 2, 14-17.

Elio, R. & Scharf, P.B. (1990). Modeling novice-to-expert shifts in problem solving strategy and knowledge organization. *Cognitive Science, 14*, 579-637.

Ferguson-Hessler, M.G. & de Jong, T. (1990). Studying physics texts: Differences in study processes between good and poor solvers. *Cognition and Instruction, 7*, 41-75.

Gilmore, D. & Self, J. (1988). The application of machine learning to intelligent tutoring systems. In J. Self (Ed.) *Artificial Intelligence and human learning, intelligent computer-aided instruction.* Chapman and Hall: New York.

Kerslake, D. (1986). *Fractions: children's strategies and errors.* NFER-Nelson, Mindsor, Berkshire.

Hart, K. & Kerslake, D. (1983). *Avoidance of fractions.* Paper presented at the Annual Meeting of the American Educational Research Association. Montreal, CA.

Hiebert, J. & Behr, M. (Eds.) (1988). *Number concepts and operations in the middle grades. Vol. 2.* Hillsdale, NJ: Erlbaum

Hintze, D. (1991) Guided discovery learning: The computer as a cooperative learning assistant. In L. Birnbaum (Ed.) *The international conference on the learning sciences: Proceedings of the 1991 conference.* Charlottesville, VA: Association for the Advancement of Computing in Education.

Holland, J., Holyoak, K., Nisbett, R., & Thagard, P. (1986). *Induction: The process of inference, learning, and discovery.* Cambridge, MA: MIT Press.

Jones, R. & VanLehn, K. (1992). A fine-grained model of skill acquisition: Fitting Cascade to individual subjects. In *Proceedings of the fourteenth annual meeting of the Cognitive Science Society*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Kieras, D.E. (1992) *Semantics-based reference resolution in technical text processing: An exploration of using the WordNet database in the Computerized Comprehensibility System.* Technical report 35. Ann Arbor, MI: Technical Communications Program, University of Michigan.

Kieras, D.E. (1991). *Human learning of schemas from explanations in practice electronics.* Technical report 34. Ann Arbor, MI: Technical Communications Program, University of Michigan.

Kieras, D.E. (1990). The role of cognitive simulation models in the development of advanced training and testing systems. In N. Frederiksen, R. Glaser, A. Lesgold & M. Shafto (Eds.), *Diagnostic monitoring of skill and knowledge acquisition.* pp. 51-74, Hillsdale, NJ: Lawrence Erlbaum Associates.

Kieras, D.E. (1989). An advanced computerized aid for the writing of comprehensible technical documents. In B. Britton & S. Glynn (Eds.), *Computer writing environments: Theory, research*

and design. Hillsdale, NJ: Lawrence Erlbaum Associates.

Komoski, K.P. (1974). An imbalance of product quantity and instructional quality: The imperative of empiricism. *Audio Visual Communication Review, 4*, 357-386.

Larkin, J. (1981). Enriching formal knowledge: A model for learning to solve textbook physics problems. In J.R. Anderson (Ed.) *Cognitive Skills and their Acquisition.* Hillsdale, NJ: Lawrence Erlbaum Associates.

LeFevre, P. (1986). *Exploring fractions with fourth graders.* Paper presented at the Annual Meeting of the American Educational Research Association. San Francisco, CA.

Lepper, M.R., Aspinwall, L., Mumme, D., & Chabay, R.W. (1990). Self-perception and social perception processes in tutoring: Subtle social control strategies of expert tutors. In J.M. Olson & M.P. Zanna (Eds.), *Self inference processes: the sixth Ontario symposium in social psychology.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Lesgold, A., Jajoie, S., Bunzo, M., & Eggan, G. (1991). SHERLOCK: A coached practice environment for an electronics troubleshooting job. In J.H. Larkin & R.W. Chabay (Eds.), *Computer assisted instruction and intelligent tutoring systems: Shared goals and complementary approaches.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Mach, N. K. (1988). *Using estimation to learn fractions with understanding.* Paper presented at the Annual Meeting of the American Educational Research Association, New Orleans, LA.

Mach, N. K. (1988). *Learning fractions with understanding: Building upon informal knowledge.* Paper presented at the Annual Meeting of the American Educational Research Association, New Orleans, LA.

Martin, J. & VanLehn, K. (1993). OLAE: Progress toward a multi-activity, Bayesian student modeler. In S. P. Brna, S. Ohlsson & H. Pain, (Eds.), *Artificial Intelligence in Education, 1993: Proceedings of AI-ED 93.* Charlottesville, VA: Associate for the Advancement of Computing in Education.

Mayer, J.H. (1990) *Explanation-based knowledge acquisition of schemas in practical electronics.* Tech. Rept. TR-90/ONR-32. Technical Information Design and Analysis Laboratory, University of Michigan, Ann Arbor, MI.

Merrill, D.C., Reiser, B.J., Ranney, M. & Trafton, J.G. (1992). Effective tutoring techniques: A comparison of human tutors and intelligent tutoring systems. *The Journal of the Learning*

Sciences, 2, 3, 277-305.

Nason, R.A.(1991). PRODIGY: An intelligent computer-based system for developing teacher's expertise in the diagnosis and remediation of common error patterns in the domain of common fractions. *Interactive Learning International, 7,* 243-252.

Nason, R.A. (1993). Prodigy: Diagnosis and remediation in the domain of common fraction. *Computers and Education: An International Journal, 20,* 1, 45-53.

Neves, D.M. (1978). A computer program that learns algebraic procedures by examining examples and working problems in a textbook. *Proceedings of the Second Biennial Conference of the Canadian Society for Computational Studies of Intelligence.* 191-195. Toronto, Ontario, Canada.

Neves, D.M. (1981). *Learning procedures from examples.* Unpublished doctoral dissertation. Dept. of Psychology, Carnegie-Mellon University, Pittsburgh, PA.

Neves, D.M. & Anderson, J.R. (1981). Knowledge compilation: Mechanisms for the automatization of cognitive skills. In J.R. Anderson (Ed.) *Cognitive skills and their acquisition.* pp. 57-84. Hillsdale, NJ: Lawrence Erlbaum Associates.

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

O'Donnell, A.M., Dansereau, D.F., Hall, R.H., 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.

Ohlsson, s. (1987). Truth versus appropriateness: Relating declarative to procedural knowledge. In D. Klahr, P. Langley & R. Neches (Eds.), *Production system models of learning and development*, Cambridge, MA: MIT Press.

Ohlsson, S., (1991) Knowledge requirement for teaching: The case of fractions. In P. Goodyear (Ed.), *Teaching knowledge and intelligent tutoring*, 25-59. Norwood, NJ: Ablex.

Ohlsson, S. (1993) The interaction between knowledge and practice in the acquisition of cognitive skills. In A. Meyrowitz & S. Chipman (Eds.), *Cognitive models of complex learning.* Norwell, MA: Kluwer Academic Publishers.

Ohlsson, S., Ernst, A.M. & Rees, E. (1992). The cognitive complexity of doing and learning arithmetic. *Journal for Research in Mathematics Education, 23,* 5, 441-467.

Ohlsson, S. & Rees, E. (1991) The function of conceptual understanding in the learning of arithmetic procedures. *Cognition and Instruction, 8,* 103-179.

Ortony, A., Clore, G.L. & Collins, A. (1988) *The cognitive structure of emotions.* New York: Cambridge University Press.

Palthepu, S., Greer, J. & McCalla, G. (1991). Learning by teaching. In L. Birnbaum (Ed.) *The international conference on the learning sciences: Proceedings of the 1991 conference.* Charlottesville, VA: Association for the Advancement of Computing in Education.

Pirolli, P. (1991). Effects of examples and their explanations in a lesson on recursion: A production system analysis. *Cognition and Instruction, 8*, 207-259.

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

Pressley, M., Wood, E., Woloshyn, V.E., Martin, V., King, A. & Menke, D. (1992). Encouraging mindful use of prior knowledge: Attempting to construct explanatory answers facilitates learning. *Educational Psychologist, 27*, 991-109.

Post, T. R., Wachsmuth, I., Lesh, R. & Behr, M.J. (1985). Order and equivalence or rational numbers: A cognitive analysis. *Journal of Research in Mathematics Educations, 16*, 1, 18-36.

Putnam, R.T. (1987). Structuring and adjusting content for students: A study of live and simulated tutoring of addition. *American Educational Research Journal, 24*, 13-48.

Redmond, M. (1989) Combining explanation types for learning by understanding instructional examples. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Redmond, M. (1992) *Learning by observing and understanding expert problem solving.* Tech. Rept. GIT-CC-992/43. Doctoral Dissertation, College of Computing, Georgia Institute of Technology, Atlanta, GA.

Resnick, L.B. & Omanson, S.F. (1987). Learning to understand arithmetic. In R. Glaser (Ed.) *Advances in Instructional Psychology, Vol. 3*, Hillsdale, NJ: Lawrence Erlbaum Assoc.

Shelley, A. & Sibert, E. (1991). Computer simulation in teacher education: Enhancing teacher decision making. In P. Goodyear (Ed.) *Teaching knowledge and intelligent tutoring.* Norwood, NJ: Ablex.

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

Swanson, J.H. (1992). *What does it take to adapt instruction to the individual?: A case study of one-to-one tutoring.* Paper presented at the annual conference of the American Educational Research Association, San Francisco, CA.

Tatsuoka, K.K., (1984) *Analysis of errors in fraction addition and subtraction problems.* Computer-based Education Research Laboratory, University of Illinois, Urbana, IL.

Twidale, M. (1993). Redressing the balance: The advantages of informal evaluation techniques for intelligent learning environments. *Journal of Artificial Intelligence in Education, 4*, 2/3, pp. 155-178.

VanLehn, K. (1988). Student modeling. In M.C. Polson & J.J. Richardson (Eds.) *Foundations of intelligent tutoring systems.* Hillsdale, NJ: Lawrence Erlbaum Associates.

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

VanLehn, K. (1988) Student modeling. In M. Polson & J. Richardson (Eds.)*Foundations of Intelligent Tutoring Systems*, Hillsdale, NJ: Erlbaum. pp. 55-78.

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

VanLehn, K. (1989b) *Mind bugs: The origins of procedural misconceptions.* Cambridge, MA: MIT Press.

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

VanLehn, K. & Jones, R.M. (1993a) Learning by explaining examples to oneself: A computational model. In A. Meyrowitz & S. Chipman (Eds.) *Cognitive Models of Complex Learning.* Boston, MA: Kluwer Academic.

VanLehn, K. & Jones, R.M. (1993b). What mediates the self-explanation effect? Knowledge gaps, schemas or analogies? *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Erlbaum.

VanLehn, K. & Jones, R.M. (1993c). Better learners use analogical problem solving sparingly. *Proceedings of the Tenth International Conference on Machine Learning.* Los Altos, CA: Morgan-Kaufman.

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

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

Wood, S. (1991). A model of classroom processes: Towards the formalization of experienced teacher's professional knowledge. In P. Goodyear (Ed.) *Teaching Knowledge and Intelligent Tutoring.* pp. 297-313. Norwood, NJ: Ablex.