Methods for Evaluating Simulated Learners: Examples from SimStudent

Kenneth R. Koedinger[1], Noboru Matsuda[1], Christopher J. MacLellan[1], and Elizabeth A. McLaughlin[1]

[1] Carnegie Mellon University, Pittsburgh, PA
koedinger@cmu.edu

Abstract. We discuss methods for evaluating simulated learners associated with four different scientific and practical goals for simulated learners. These goals are to develop a precise theory of learning, to provide a formative test of alternative instructional approaches, to automate authoring of intelligent tutoring systems, and to use as a teachable agent for students to learn by teaching. For each goal, we discuss methods for evaluating how well a simulated learner achieves that goal. We use SimStudent, a simulated learner theory and software architecture, to illustrate these evaluation methods. We describe, for example, how SimStudent has been evaluated as a theory of student learning by comparing, across four domains, the cognitive models it learns to the hand-authored models. The SimStudent-acquired models generally yield more accurate predictions of student data. We suggest future research into directly evaluating simulated learner predictions of the process of student learning.

Keywords: cognitive models, learning theory, instructional theory

1 Introduction

When is a simulated learner a success? We discuss different approaches to evaluating simulated learners (SLs). Some of these evaluation approaches are technical in nature, whether or how well a technical goal has been achieved, and some are empirical, whereby predictions from the SL are compared against data. These approaches can be framed with respect to four goals for developing SLs (see Table 1). These goals have been pursued in prior SL research, such as the use of "pseudo-students" [1] to test the quality of an instructional design (#2 in Table 1). Before describing different evaluation approaches appropriate for different goals, we first introduce SimStudent.

1.1 SimStudent: A Simulated Learner Theory and Software Architecture

SimStudent [2,3] is an SL system and theory in the class of adaptive production systems as defined by [4]. As such, it is similar to cognitive architectures such as ACT-R [5], Soar [6], and Icarus [7], however, it distinctly focuses on modeling inductive knowledge-level learning [8] of complex academic skills learning. SimStudent learns

from a few primary forms of instruction, including examples of correct actions, skill labels on similar actions, clues for what information in the interface to focus on to infer a next action, and finally yes-or-no feedback on actions performed by SimStudent.

Table 1. Scientific and Practical Goals for Simulated Learners (SLs)

1. *Precise Theory*. Use SLs to develop and articulate precise theory of learning.
   a. *Cognitive Model*. Create theories of domain expertise
   b. *Error Model*. Create theories of student domain misconceptions
   c. *Prior Knowledge*. Create theories of how prior knowledge changes learning
   d. *Learning Process*. Create theories of change in knowledge and performance
2. *Instructional Testing*. Use SLs as a "crash test" to evaluate instruction
3. *Automated Authoring*. Use SLs to automatically an intelligent tutoring system
4. *Teachable Agent*. Use SLs as a teachable agent or peer

To tutor SimStudent, a problem is entered in the tutoring interface (e.g., 2x = 8 in row 1 of Figure 1). SimStudent attempts to solve the problem by applying productions learned so far. If an applicable production is found, it is fired and problem interface is updated. The author then provides correctness *feedback* on SimStudent's step. If no correct production application is found, SimStudent asks the author to demonstrate the next step directly in the interface. When providing a demonstration, the author first specifies the *focus of attention* (i.e. input fields relevant to the current step) by double-clicking the corresponding interface elements (e.g., the cells containing 2x and 8 in Figure 1). The author takes action using the relevant information (e.g., entering divide 2 in Figure 1). Finally, the *author specifies a skill name* by clicking on the newly added edge of the behavior graph. This skill label is used to help guide SimStudent's learning and to make production rule names more readable.

SimStudent uses three machine-learning mechanisms (*how*, *where*, and *when*) to acquire production rules. When given a new demonstration (i.e., a positive example of a rule), SimStudent uses its *how* learner to produce a general composition of functions that replicate the demonstrated steps and ones like it. For example, in Figure 1, when given the demonstration "divide 2*"* for the problem 2x=8, SimStudent induces that the result of the "get-first-integer-without-sign" function when applied to left side of the problem and appended to the word "divide" explains the demonstration.

After an action sequence has been discovered, SimStudent uses its *where* learner to identify a generalized path to the focus of attention in the tutor interface. In Figure 1, the *where* learner discovers retrieval paths for the three cells in the first column. These paths are generalized as more positive examples and are acquired for a given rule. For example, when the author demonstrates the application of the divide rule shown in Figure 1 to the second row of the equation table, then the production retrieval path is generalized to work over any row in the equation table.

Finally, after learning an action sequence and general paths to relevant information, SimStudent uses its *when* learner to identify the conditions under which the learned production rule produces correct actions. For example, in Figure 1 SimStudent learns that this rule can only be correctly applied when one side of the equation

has a coefficient. In situations when SimStudent receives positive and negative feedback on its rule applications, it uses the *when* learner to update the conditions on the rules. Note, the *how* and *where* learners primarily use positive examples.

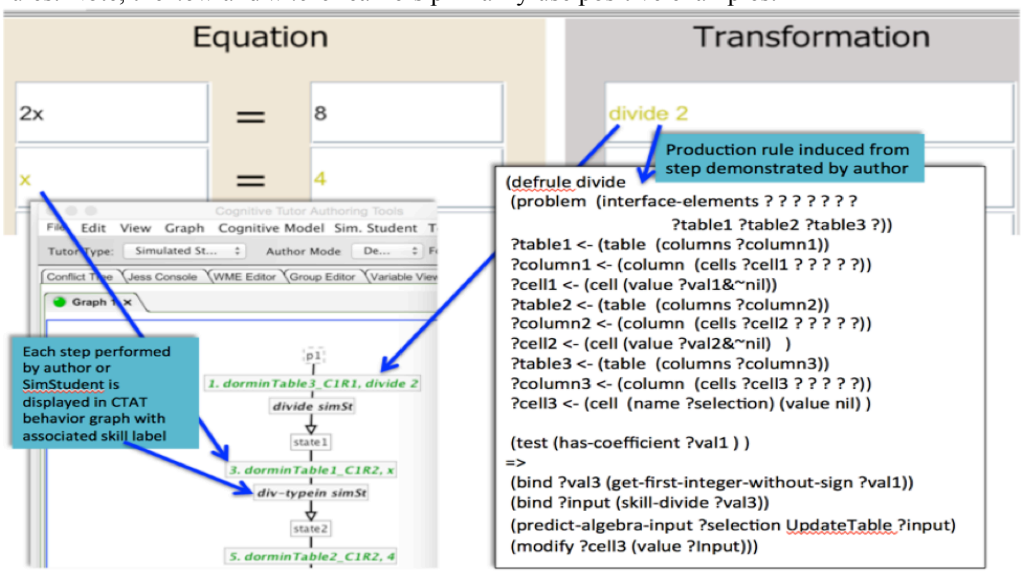

Fig. 1. After entering a problem, "2x=8" (top left), teaching of SimStudent occurs either by giving yes-or-no feedback when SimStudent attempts a step or by demonstrating a correct step when SimStudent cannot (e.g., "divide 2").

SimStudent is also capable of learning the representation of the chunks that make up the production system's working memory and are the informational basis on which productions are learned. It does so using an unsupervised grammar induction approach [3]. This feature sets it apart from other production rule learning systems.

2 Evaluating Simulated Learners as Theories of Learning

It is helpful to distinguish a general theory of learning from a theory of *student* learning. We focus on student learning because of the goals of AI in Education. However, it is worth mentioning evaluation criteria for a general learning theory, such as how quickly and independently learning takes place and how general and accurate is resulting performance. These criteria facilitate comparative evaluations. For instance, hierarchical Bayesian models are arguably better models of learning than other classification or neural network models because they learn as well with fewer examples [9].

2.1 Good Learning Theory Should Generate Accurate Cognitive Models

A student learning theory should produce the kind of expertise that human students acquire. In other words, the result of teaching an SL should be a cognitive model of

what human student's know after instruction. Thus, one way to evaluate an SL is to evaluate the quality of the cognitive models it produces. We proposed [10] six constraints to evaluate the quality of a cognitive model: 1) solution sufficiency, 2) step sufficiency, 3) choice matching, 4) computational parsimony, 5) acquirability, and 6) transfer. The first two are empirical and qualitative: Is the cognitive model that the SL acquires able to solve tasks and do so with steps that are consistent with human students? The third is quantitative: Does the frequency of strategy use and common error categories generated by the cognitive model on different tasks correspond with the same frequencies exhibited by human students? The last three are rational in character, involving inspection of the cognitive model.

These constraints were designed with hand-authored models in mind, so some, like the acquirability constraint (#5), appear trivial in the SL context. There is no question that the components of an SL-produced cognitive model are plausibly acquired because the SL does, in fact, acquire them. Similarly, the solution sufficiency constraint (#1) is straightforwardly achieved if the SL does not indeed succeed in learning the task domain. If the cognitive model that is produced solves problems using the kinds of intermediate steps used in student solutions, for example, it performs its solution in a step-based tutoring system interface, then the step sufficiency constraint (#2) is met.

How, then, can the remaining constraints be evaluated? In [11], we employed an educational data mining approach that evaluates the accuracy of a cognitive model by a "smooth learning curve" criteria [cf., 12,13]. Using a relatively simple statistical model of how instructional opportunities improve the accuracy of knowledge, this approach can measure and compare cognitive models in terms of their accuracy in predicting learning curve data. To employ the statistical model fit, the cognitive model is simplified into a "Q matrix", which maps each observed task performed (e.g., entering a step in a problem solving) to the knowledge components hypothesized to be needed to successfully perform that task. For any appropriate dataset uploaded into DataShop (learnlab.org/DataShop), the website allows users to edit and upload alternative cognitive models (in the Q matrix format), automatically performs statistical model fits, renders learning curve visualizations, and displays a rank ordering of the models in terms of their predictive accuracy [14].

We used this approach to evaluate the empirical accuracy of the cognitive models that SimStudent learns as compared to hand-authored cognitive models [11]. SimStudent was tutored in four domains: algebra, fractions, chemistry, and English grammar, in which we had existing human data and existing hand-authored cognitive models. In each domain SimStudent induced, from examples and from practice with feedback, both new chunk structures to represent the organization (or "grammar") of the perceptual input and new production rules that solve problems (e.g., add two fractions) or make decisions (e.g., select when to use "the" or "a" in English sentences). In each case, the production rules that SimStudent acquired were converted into the Q matrix format. Then the DataShop cognitive model comparison was employed to compare whether these models fit student learning curve data better than the hand-authored cognitive models do.

In all four domains, the SimStudent-acquired cognitive models made distinctions not present in the hand-authored models (e.g., it had two different production rules

across tasks for which the hand-authored model had one) and thus it tended to produce models with more knowledge components (as shown in Table 2). For example, SimStudent learned two different production rules for the typical last step in equation solving where one production covered typical cases (e.g., from 3x = 12 the student should "divide by 3") and another covered a perceptually distinct special case (e.g., from -x = 12 the student should divide by -1).

In all four domains, at least some of these distinctions improved the predictive fit to the learning curve data for the relevant tasks. For example, the SimStudent-acquired cognitive model in algebra leads to better accuracy because real students had a much higher error rate on tasks like -x=12 (where the coefficient, -1, is implicit) than on tasks like 3x=12 (where the coefficient, 3, is explicitly visible). In one domain (Fraction Addition,), the SimStudent-acquired cognitive model failed to make a key distinction present in the hand-authored model and thus, while better in some cases, its overall fit was worse. In the three other domains, the SimStudent-acquired cognitive models were found to be more accurate than the hand-authored cognitive models.

Table 2. A comparison of human-generated and SimStudent-discovered models.

	Number of Production Rules		Cross-Validated RMSE	
	Human-Generated Model	SimStudent Discovered Model	Human-Generated Model	SimStudent Discovered Model
Algebra	12	21	0.4024	**0.3999**
Stoichiometry	44	46	0.3501	**0.3488**
Fraction Addition	8	6	**0.3232**	0.3343
Article selection	19	22	0.4044	**0.4033**

In other words, this "smooth learning curve" method of evaluation can provide evidence that an SL, SimStudent in this case, is a reasonable model of student learning in that it acquires knowledge at a grain size (as represented in the components of the cognitive model) that is demonstrably consistent with human data.

One limitation of this approach is that it *indirectly* compares an SL to human learners through the process of fitting a statistical model. In the case of algebra, for example, SimStudent's acquisition of two different productions for tasks of the form Nx=N versus tasks of the form -x=N gets translated into a prediction that student performance will be *different* in these situations, but the not direction of the difference. The parameter estimation in statistical model fit yields the prediction for which of these task categories (Nx=N or -x=N) is harder. A more *direct* comparison would not use an intermediate statistical model fit. It would require the SL to not only produce a relevant distinction, but to make a prediction of student performance differences, such as whether it takes longer to successfully learn some kinds of tasks than others. Such an evaluation approach is discussed in section 2.3.

2.2 Matching Student Errors and Testing Prior Knowledge Assumptions

As a model of student learning, a good SL should not only produce accurate performance with learning, but should also produce the kinds of errors that students produce [cf.,15]. Thus, comparing SL errors to student errors is another way to evaluate an SL.

One theory of student errors is that students learn incorrect knowledge (e.g., incorrect production rules or schemas) from *correct example-based instruction* due to the necessary fallibility of inductive learning processes. A further hypothesis is that inductive learning errors are more likely when students have "weak" (i.e., more domain general) rather than "strong" (i.e., more domain specific) prior knowledge. With weak prior knowledge, students may interpret examples shallowly, paying attention to more immediately perceived surface features, rather than more deeply, by making domain-relevant inferences from those surface features. Consider example-based instruction where a student is given the equation "3x+5 = 7" and told that "subtract 5" from both sides is a good next step. A novice student with weak prior knowledge might interpret this example shallowly, as subtracting a number (i.e., 5) instead of more deeply, as subtracting a term (i.e., +5). As a consequence, the student may induce knowledge that produces an error on a subsequent problem, such as "4x-2=5" where they subtract 2 from both sides. Indeed, this error is common among beginning algebra students.

We evaluated SimStudent by comparing induction errors it makes with human student errors [16]. More specifically, we evaluated the weak prior knowledge hypothesis expressed above. We conducted a simulation study by having multiple instances of SimStudent get trained by the Algebra Cognitive Tutor. We compared SimStudent behaviors with actual student data from the Cognitive Tutor's logs of student interactions with the system. When SimStudent starts with weak prior knowledge rather than strong prior knowledge, it learns more slowly, that is, the accuracy of learned skills is lower given the same amount of training. More importantly, SimStudent's ability to predict student errors increased significantly when given weak rather than strong prior knowledge. In fact, the errors generated by SimStudent with strong prior knowledge were almost never the same kinds of errors commonly made by real students.

In addition to illustrating how an SL can be evaluated by comparing its error generation to human errors, this example illustrates how an SL can be used to test assumptions about student prior knowledge. In particular, SimStudent provides a theoretical explanation of empirical results [17] showing correlations between tasks measuring prior knowledge (e.g., identify the negative terms in "3x-4 = -5-2x") and subsequent learning of target skills (e.g., solving algebra equations).

Some previous studies of students' errors focus primarily on a descriptive theory to explain why students made particular errors, for example, repair theory [15], the theory of bugs [18], and the theory of extrapolation technique [19]. With SLs, we can better understand the process of acquiring the incorrect skills that generate errors. The precise understanding that computational modeling facilitates provides us with insights into designing better learning environments that mitigate error formation.

2.3 Good Student Learning Theory Should Match Learning Process Data

Matching an SL's performance to learning process data is similar to the cognitive model evaluation discussed above in section 2.1. However, as indicated above, that approach has the limitation of being an indirect comparison with human data whereby there the fit to human data is, in a key sense, less challenging because it is mediated by a separate step parameter estimation of a statistical model. A more direct compari-

son is, in simple terms, to match the behavior of multiple instances of an SL (i.e., a whole simulated class) with the behavior of multiple students. The SLs interact with a tutoring system (like one shown in Figure 2) just as a class of human students would and their behavior is logged just as human student data is. Then the simulated and human student data logs can be compared, for example, by comparing learning curves that average across all (simulated and human) student participants.

3 Evaluating Simulated Learners as Instruction Testers

A number of projects have explored the use of an SL to compare different forms of instruction. VanLehn was perhaps the first to suggest such a use of a "pseudo student" [1]. A version of ACT-R's utility learning mechanism was used to show that the SL was more successful when given error feedback not only on target performance tasks (e.g., solving two-step equations), but also on shorter subtasks (e.g., one-step equations) [10]. A SimStudent study showed better learning from a combination of examples and problems to solve, than just giving it examples [2]. Another showed that interleaving problem types is better for learning than blocking problem types because interleaving provides better opportunities correcting over-generalization errors [20].

For a general theory of instruction, it is of scientific interest to understand the effectiveness of different forms of instruction for different kinds of SL systems even if the SL is not an accurate model of student learning. Such understanding is relevant to advancing applications of AI and is directly relevant to using an SL for automated ITS authoring (next section). Such theoretical demonstrations may also have relevance to a theory of *human* instruction as they may 1) provide theoretical explanations for instructional improvements that have been demonstrated with human learners or 2) generate predictions for what may work with human students.

These instructional conclusions can only be reliably extended to human learners when the SL is an accurate model of student learning. The most reliable evaluation of an SL as instructional tester is a follow-up random assignment experiment with human learners that demonstrates that the instructional form that was better for the SLs is also better for students. In the examples given above, there is some evidence that the SLs are accurate models of student learning (e.g., past relevant human experiments). However, in none of them was the ideal follow-up experiment performed.

4 Evaluating Simulated Learners as ITS Authoring Tools

In addition to their use as theories of learning and for testing instructional content, simulated learning systems can also be used to facilitate the authoring of Intelligent Tutoring Systems (ITS). In particular, once an SL has been sufficiently trained, the cognitive model it learns can then be used directly as an expert model. Previous work, such as Example Tracing tutor authoring [21], has explored how models can be acquired by demonstration. However, by using a simulated learning system to induce general rules form the demonstrations more general models can be acquired more efficiently. For example, the use of SimStudent as authoring tool is still experimental,

but there is evidence that it may accelerate the authoring process and produce more accurate cognitive models than hand authoring. One demonstration explored the benefits of a traditional programming by demonstration approach to authoring in SimStudent versus a programming by tutoring approach [2]. In the latter, SimStudent asks for demonstrations only at steps where it has no relevant productions. Otherwise, it performs a step and asks the author for feedback as to whether the step is correct or not. Programming by tutoring was found to be much faster than programming by demonstration (77 minutes vs. 238 minutes) and produced a more accurate cognitive model whereby there were fewer productions that produced over-generalization errors. Programming by tutoring is now the standard approach because of its improved efficiency and effectiveness. Better efficiency is obtained because many author demonstrations are replaced by SimStudent actions with a quick yes-or-no response. Better effectiveness is obtained because these actions expose over-generalization errors to which the author responds "no" and the system learns new if-part preconditions to more appropriately narrow the generality of the modified production rule.

A second demonstration of SimStudent as an authoring tool [22] compared authoring in SimStudent with authoring example-tracing tutors in CTAT. Tutoring SimStudent has considerable similarity with creating an example-tracing tutor except that SimStudent starts to perform actions for the author, which can be merely checked as desirable or not, saving the time it otherwise takes for an author to perform those demonstrations. This study reported a potential savings of 43% in authoring time.

5 Evaluating a Simulated Learner as a Teachable Agent

Simulated learner systems can be more directly involved in helping students learn when they are used as a teachable agent whereby students learn by teaching [cf., 23]. Evaluating the use of an SL in this form ideally involves multiple steps. One should start with an SL that has already received some positive evaluation as a good model of student learning (see section 2). Then incorporate it into a teachable agent architecture and, as early and often as possible, perform pilot students with individual students [cf., 24 on think aloud user studies) and revise the system design. Finally, for both formative and summative reasons, use random assignment experiments to compare student learning from the teachable agent with reasonable alternatives.

Using SimStudent, we built a teachable agent environment, called APLUS, in which students learn to solve linear equations by teaching SimStudent [25]. To evaluate the effectiveness of APLUS and advance the theory of learning by teaching, we conducted multiple *in vivo* experiments [25,26,27,28]. Each of the classroom studies have been randomized controlled trials with two conditions varying one instructional approach. In one study [25], the self-explanation hypothesis was tested. To do so, we developed a version of APLUS in which SimStudent occasionally asked "why" questions. For example, when a student provided negative feedback to a step SimStudent performed, SimStudent asked, "Why do you think adding 3 here on both sides is incorrect?" Students were asked to respond to SimStudent's questions either by selecting pre-specified menu items or entering a free text response. The results showed that

the amount and the level of elaboration of the response had a reliable correlation with students' learning measured by online pre- and post-tests.

6 Conclusion

We outlined four general purposes for simulated learners (see Table 1) and reviewed methods of evaluation that align with these purposes. To evaluate an SL as a precise theory of learning, one can evaluate the cognitive model that results from learning, evaluate the accuracy of error predictions as well as prior knowledge assumptions needed to produce those errors, or evaluate the learning process, that is, the changes in student performance over time. To evaluate an SL as an instructional test, one should not only evaluate the SL's accuracy as a theory of student learning, but should also perform human experiments to determine whether the instruction that works best for SLs also works best for human students. To evaluate an SL as an automated authoring tool, one can evaluate the speed and precision of rule production, the frequency of over-generalization errors and the fit of the cognitive models it produces. More ambitiously, one can evaluate whether the resulting tutor produces as good (or better!) learning than an existing tutor. Similarly, to evaluate an SL as a Teachable Agent, one can not only evaluate the system features, but also perform experiments on whether students learn better with that system than with reasonable alternatives.

Simulated learner research is still in its infancy so most evaluation methods have not been frequently used. We know of just one such study [29] that evaluated an SL as an instructional tester by following up a predicted difference in instruction with a random assignment experiment with real students. It used an extension of the ACT-R theory of memory to simulate positive learning effects of an optimized practice schedule over an evenly spaced practice schedule. The same experiment was then run with human students and it confirmed the benefits of the optimized practice schedule. Such experiments are more feasible when the instruction involved is targeting simpler learning processes, such as memory, but will be more challenging as they target more complex learning processes, such as induction or sense making [31].

The space of instructional choices is just too large, over 200 trillion possible forms of instruction [32], for a purely empirical science of learning and instruction to succeed. We need parallel and coordinated advances in theories of learning *and* instruction. Efforts to develop and evaluate SLs are fundamental to such advancement.

References

1. VanLehn (1991). Two pseudo-students: Applications of machine learning to formative evaluation. In Lewis & Otsuki (Eds.), *Adv Res on Comp in Ed* (pp. 17-26). Amsterdm: Els.
2. Matsuda, Cohen, Koedinger (2015). Teaching the teacher. *Int J of AI in Ed, 25*, 1-34.
3. Li, Matsuda, Cohen, Koedinger (2015). Integrating representation learning and skill learning in a human-like intelligent agent. *AI, 219,* 67-91.
4. Anzai, Y. & Simon (1979). The theory of learning by doing. *Psych Rev, 86 (2)*, 124-140.
5. Anderson, J.R., & Lebiere, C. (1998). *The Atomic Components of Thought*. Hillsdale: Erl.

6. Laird, Newell, & Rosenbloom (1987). Soar. *AI, 33*(1), 1–64.
7. Langley & Choi (2006). A unified cognitive architecture for physical agents. In *Proc of AI*.
8. Newell, Allen. 1990. *Unified Theories of Cognition*. Cambridge, MA: Harvard U. Press.
9. Tenenbaum, J. B., Griffiths, T. L., & Kemp.C. (2006). Theory-based Bayesian models of inductive learning and reasoning. *Trends in Cognitive Sciences*, *10*, 309-318.
10. MacLaren & Koedinger (2002). When and why does mastery learning work: Instructional experiments with ACT-R "SimStudents". In *Proc of ITS*, 355-366. Berlin: Spr-Ver.
11. Li, Stampfer, Cohen, & Koedinger (2013). General and efficient cognitive model discovery using a simulated student. In *Proc of Cognitive Science*. (pp. 894-9) Austin, TX.
12. Martin, Mitrovic, Mathan & Koedinger (2011). Evaluating and improving adaptive educational systems with learning curves. *User Modeling and User-Adapted Int, 21*(3), 249-283.
13. Stamper, J.C. & Koedinger, K.R. (2011). Human-machine student model discovery and improvement using data. In *Proc of AI in Ed*, pp. 353-360. Berlin: Springer.
14. Koedinger, Baker, Cunningham, Skogsholm, Leber, Stamper, (2010). A Data Repository for the EDM community: The PSLC DataShop. In *Hdbk of Ed Data Min*. Boca Rat: CRC.
15. Brown, J. S., & VanLehn, K. (1980). Repair theory. *Cognitive Science, 4,* 379-426.
16. Matsuda, Lee, Cohen, & Koedinger, (2009). A computational model of how learner errors arise from weak prior knowledge. In *Proc of Cognitive Science*. pp. 1288-1293.
17. Booth, J.L., & Koedinger, K.R. (2008). Key misconceptions in algebraic problem solving. In Love, McRae & Sloutsky (Eds.), *Proc of Cognitive Science,* pp. 571-576.
18. VanLehn, K. (1982). Bugs are not enough. *Journal of Mathematical Behavior, 3(2)*, 3-71.
19. Matz, M. (1980). Towards a process model for high school algebra errors. In Sleeman & Brown (Eds.), *Intelligent Tutoring Systems* (pp. 25-50). Orlando, FL: Academic Press.
20. Li, N., Cohen, W. W., & Koedinger, K. R. (2012). Problem Order Implications for Learning Transfer. In *Proceedings of Intelligent Tutoring Systems*, 185–194.
21. Aleven, V., McLaren, B., Sewall, J., & Koedinger, K. R. (2009). Example-tracing tutors: A new paradigm for intelligent tutoring systems. *Int J of AI in Education, 19,* 105-154.
22. MacLellan, Koedinger & Matsuda (2014). Authoring Tutors with SimStudent: An Evaluation of Efficiency and Model Quality. *Proc of Intelligent Tutoring Systems,* 551-560.
23. Biswas, G., Schwartz, D., Leelawong, K., Vye, N. (2005). Learning by Teaching: A New Agent Paradigm for Educational Software. *Applied Artificial Intelligence, 19*, 363-392.
24. Gomoll, K., (1990). Some techniques for observing users. In Laurel B. (ed.), *The Art of Human-Computer Interface Design*, Addison-Wesley, Reading, MA, pp. 85-90.
25. Matsuda, Yarzebinski, Keiser, Raizada, William, Stylianides & Koedinger (2013). Cognitive anatomy of tutor learning. *J of Ed Psy, 105(4)*, 1152-1163.
26. Matsuda, Cohen, Koedinger, Keiser, Raizada, Yarzebinski, Watson, Stylianides (2012). Studying the effect of tutor learning using a teachable agent. In *Proc of DIGITEL*, 25-32.
27. Matsuda, Griger, Barbalios, Stylianides, Cohen, & Koedinger (2014). Investigating the effect of meta-cognitive scaffolding for learning by teaching. In *Proc of ITS,* 104-113.
28. Matsuda, Keiser, Raizada, Tu, Stylianides, Cohen, Koedinger (2010). Learning by Teaching SimStudent: In *Proceedings of Intelligent Tutoring Systems,* 317-326.
29. Ritter, Anderson, Koedinger, & Corbett (2007). Cognitive tutor: Applied research in mathematics education. *Psychonomic Bulletin &Review, 14(2)*, 249-255.
30. Pavlik & Anderson (2008). Using a model to compute the optimal schedule of practice. *Journal of Experimental Psychology: Applied, 14*, 101-117.
31. Koedinger, Corbett, Perfetti (2012). The KLI framework. *Cog Sci, 36 (5),* 757-798.
32. Koedinger, Booth, Klahr (2013). Instructional complexity. *Science, 342*, 935-937.