Submitted to CogSci09

A Computational Model of How Learner Errors Arise from Weak Prior Knowledge

Noboru Matsuda (Noboru.Matsuda@cs.cmu.edu)
Andrew Lee (AndrewLee@cmu.edu)
William W. Cohen (WCohen@cs.cmu.edu)
Kenneth R. Koedinger (Koedinger@cmu.edu)
School of Computer Science, Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213 USA

Abstract

How do differences in prior conceptual knowledge affect the nature and rate of learning? To answer this question, we built a computational model of learning, called SimStudent, and conducted a controlled simulation study to investigate how learning a complex skill changes when the system is given "weak" domain-general vs. "strong" domain-specific prior knowledge. We measured learning outcomes with the rate of learning, the accuracy of learned skills (test scores), and the accuracy in predicting patterns of real student errors. We found not only that the accuracy of learned skills decreases when weak prior knowledge is given, but also the learning rate significantly slows down. The accuracy of predicting student errors also increased significantly, and SimStudent with the weak prior knowledge made the same errors that real students commonly make. These modeling results help explain empirical results connecting prior knowledge and student learning (Booth & Koedinger, 2008).

Keywords: Computational model of learning; machine learning; SimStudent; weak prior knowledge; patterns of student errors; mathematics education.

Introduction

In this paper, we present an innovative application of a synthetic student for modeling the error-prone process of student learning in a complex problem-solving domain.

Previous studies have shown that student misconceptions or flaws in their prior knowledge not only directly cause errors in reasoning (VanLehn & Jones, 1993), but may affect the learning process. For instance, Booth and Koedinger (2008) demonstrated that particular limitations in prior knowledge (e.g., treating terms in an equation as though terms and numbers were equivalent concepts) were correlated with particular strategic errors later in instruction (e.g., subtracting 4 from both sides of x-4=13). The presumed causal connection is that the nature of student prior knowledge changes the learning process and thus leads to differences in the problem-solving knowledge that is acquired. But what is this learning process and how is it affected by differences in prior knowledge?

A classic result from Chi, Feltovich, and Glaser's study (1981) that experts categorize problems with deep solution-relevant features while novices categorize problems with shallow, perceptually apparent, features is also relevant to our endeavor. We are ultimately interested in understanding how a novice goes from only being aware of shallow features to learning to encode problems in terms of deep features. Our strategy toward tackling this important question is to create a computational model of the learning process in complex math and science domains and to use fine-grain data from student learning over time to constrain model development. Our first steps involve demonstrating how a computational model of learning can learn when given shallow (or "weak") knowledge, how such learning is slower than when deep (or "strong") knowledge is available, and how learning based on shallow/weak knowledge better prior predicts patterns of real student errors.

In this study, we focus on the process of learning problem-solving skills from examples, where students generalize examples to inductively learn skills to solve problems. We are particularly interested in errors that are made by applying incorrect skills, and our computational model explains the processes of learning such incorrect skills as incorrect generalization of examples. A number of models of student errors have been proposed (Brown & Burton, 1978; Langley & Ohlsson, 1984; Sleeman, Kelly, Martinak, Ward, & Moore, 1989; Weber, 1996; Young & O'Shea, 1981). Our effort builds on past work by exploring how differences in prior knowledge change the nature of the incorrect skills acquired and make different predictions about student errors.

We hypothesize that incorrect generalizations are more likely when students have weaker, more general prior knowledge for encoding incoming information. This knowledge is typically more "shallow" or perceptually grounded and is in contrast to deeper or more abstract encoding knowledge. An example of such perceptually grounded prior knowledge is to recognize 3 in x/3 simply as a number instead of as a denominator. Such an interpretation might lead students to learn an inappropriate generalization such as "multiply both sides by a number in the left hand side of the equation" after observing x/3=5 gets x=15. If this generalization gets applied to an equation like 4x=2, the error of multiplying both sides by 4 is produced. We call this type of perceptually grounded prior knowledge "weak" prior knowledge in a similar sense as Newell and Simon's weak reasoning methods (1972). Weak knowledge can apply across domains and can yield successful results prior to domain-specific instruction. However, in contrast to "strong" domain-specific knowledge, weak knowledge is more likely to lead to incorrect conclusions.

The goal of the present paper is to investigate an impact of priot knowledge on learning problem-solving skills. We have implemented the proposed learning model as an interactive machine-learning agent, called *SimStudent* that learns skills through tutored problem-solving. To test the hypothesis about the impact of "weak" prior knowledge on learning, we conducted a controlled simulation study by giving SimStudents different types of prior knowledge and measuring learning outcomes as well as a fit to human students' error patterns.

In the rest of the paper, we first analyze typical errors that human students commonly make. The analysis is based on student-tutor interaction log data collected from a classroom study. We then provide a brief overview of SimStudent, mostly focused on its learning algorithms to present how prior knowledge affects the SimStudent learning. Finally, we describe an empirical simulation study to test our hypotheses where SimStudents are trained with different kinds of prior knowledge to measure the impact of prior knowledge on learning outcome.

Student Errors

For the current study, we used a dataset collected from a classroom study where students learned Algebra I with a commercially available Cognitive Tutor (called the Algebra Tutor hereafter) developed by Carnegie Learning Inc. The classroom study was conducted to investigate how students' prior knowledge affect the way students develop misconceptions (Booth & Koedinger, 2008).

While students were learning equation solving with the Algebra Tutor, the interaction between the individual students and the Algebra Tutor was recorded and stored in a free, open-resource repository, called DataShop (Koedinger, Cunningham, Skogsholm, & Leber, 2008) that shares experimental data collected from *in vivo* studies conducted in LearnLab participating schools maintained by the Pittsburgh Science of Learning Center (www.learnlab.org). This section describes the student-tutor interaction log data used and analysis of errors made by students.

Data

There were 71 students involved in the classroom study. A total of 19,683 transactions between the students and the Algebra Tutor were recorded. A transaction represents either (1) a student attempt at a step with possible feedback from the Tutor, or (2) a student's request for a hint with the actual hint message provided by the Tutor. Students had to perform a step correctly to proceed to the next step. Students could make multiple mistakes. They could also ask for a hint when they could not perform a step correctly. The Tutor first provided an abstract hint, but then students could have asked for a more detailed hint if necessary, until the Tutor finally provides very specific instruction on what to do next (e.g., "enter 3*x* in the highlighted cell"), the so-called "bottom-out hint."

The transactions in which students made an attempt at a step were coded as "Correct" if the attempt was recognized as a correct step by the Tutor, "Bug" if the attempt was recognized by the Tutor as known error category, or "Error" otherwise. There were a total of 11040 "Correct" transactions, 2010 "Bug" transactions, and 1097 "Error" transactions in the dataset. The remaining transactions were hint requests.

Table 1: The three most common error schemata. The problem schema is an abstracted form of an equation with A, B, and C representing numbers and *v* representing a variable. An error schema represents the error pattern by using letters from the problem schema

Error Schema	Frequency	Problem Schema
multiply by A	73	A/*v*=C, A/*v*=-C, -C=A/*v*, C=A/*v*, A*v*=C, *v*/-A=-C, C=*v*/-A, …
divide by A	42	-A*v*=C, -A*v*=-C, C=-A*v*, -C=-A*v*, *v*/A=-C, C=*v*/A, …
add –B	32	C=-B+A*v*, -B+(-A*v*)=-C, C=A*v*+(-B), -B+(-A*v*)=C, -C=-B+A*v*

Error Analysis

To analyze errors made by students, we categorized the Bug and Error transactions (a total 3107 of transactions) by abstracting the equation in the step on which an error was made and the error itself.

We abstracted an error and the equation on which the error occurred by simply replacing numbers and variables with letters. For example, when a student made an error to "multiply by 3" for "3/*x*=-4," the equation was represented as "A/*v* = -B" and the error was represented as "multiply by A." We call the abstracted form of error and problem the Error Schema and Problem Schema. Table 1 shows the three most common error schemata observed in the dataset.

SimStudent

SimStudent is an application of *programming by demonstration* (Cypher, 1993) with an underling *inductive logic programming* technique (Muggleton, 1991) that generalizes examples on correct and incorrect skill applications to learns how to apply individual skills to perform steps to solve problems.

For SimStudent, generalization for a particular skill application is done by providing a pragmatic explanation on "when" the skill should be applied on "what" part of the problem and "how" a step is made. A generalization of a skill application is then represented in the form of *production rule*. The when-part of an explanation composes the condition part (left-hand side) of the production rule. The what- and how-parts compose the action part (right-hand side) of the production rule.

Learning Algorithms

SimStudent uses two types of prior knowledge: *feature predicates* and *operators*. *Feature predicates* are boolean predicates used to test whether particular condition holds in a given situation. For example, the feature predicate isPolynomial(P) returns the boolean value true when P is a polynomial expression. Feature predicates are used to compose feature tests in the left-hand side of the production rules that SimStudent learns. *Operators* are used to compose a right-hand side operator sequence to generate the target next step in an example. For example, the operator getCoefficient(T) returns a coefficient of the term T when T is a variable term.

During tutoring, SimStudent accumulates positive examples of skill applications when (1) SimStudent correctly applies a step, or (2) the Tutor provides a "bottom-out" hint on a relevant step by giving an example of what to do. On the other hand, SimStudent accumulates negative examples for a skill application when (1) SimStudent applies the skill incorrectly and gets negative feedback from the Tutor, or (2) when a tutor provides a hint on a different skill, the context where that skill was applied becomes an implicit negative example for all other skills.

Manipulating Prior Knowledge

Students often make errors by treating numbers and variables superficially without taking the surrounding context into account. For example, when a student says $3x+2$ becomes $5x$, he/she may have added 3 and 2 to get 5 and concatenated x to it. Such behavior can be explained as if the student had recognized the tokens 3 and 2 in the expression as *numbers* and since there is a "+" in between, the student adds these numbers together.

The error analysis mentioned in the previous section showed that indeed, many of the common errors made by students can be explained in this way. Namely, students often rely exclusively on "shallow" features that are more directly perceived in the input rather than taking the broader context into account to infer a deep feature. An example of use of shallow features is the mental equivalent of "*to get a number in front of a variable*" instead of "to get a coefficient of a variable term".

Such shallow features can be modeled as prior knowledge that syntactically process domain entities much like string processing rather than relying on domain specific parsing and semantic manipulations. We call such prior knowledge *weak prior knowledge* as opposed to "strong" domain-specific prior knowledge.

In general, a particular example can be modeled both with weak and strong operators. For example, suppose a step $x/3=5$ gets demonstrated to "multiply by 3." Such step can be explained by a strong operator getDenominator($x/5$), which returns a denominator of a given fraction term and multiply that number to both sides. On the other hand, the same step can be explained by a weak operator getNumberStr($x/5$), which returns the left-most number in a given expression. In this context, the operator getNumberStr() is considered to be *weaker* than the operator getDemonimator(), because a single production rule leaned with getNumberStr() could explain broader errors. For example, imagine how we could model the error schema for "multiply by A." This error schema can be modeled with getNumberString() and multiply() – get a number and multiply both sides by that number. Without the weak operator, we need to have different (disjunctive) production rules to model the same error schema for different problem schemata – getNumerator() for A/v=C and getCoefficient() for Av=C.

Based on above observations, we have hypothesized that we would be able to simulate how students' learning incorrect skills from tutored problem-solving by providing SimStudent with weak operators. The next section describes an empirical study to test this hypothesis.

Error Analysis Study

Method

SimStudent was tutored on how to solve linear equation by interacting with Carnegie Learning Algebra I Tutor like human students learn with the Tutor interactively. That is, SimStudent was posed a problem and asked to solve it. When SimStudent performed a step, the Tutor provided flagged feedback on the correctness of the step performed. SimStudent attempted to apply rules until a step is performed correctly. If SimStudent failed to perform a step correctly, then SimStudent asked the Tutor for a hint. The Tutor then provided a bottom out hint by demonstrating how to perform the step.

There were two experimental conditions: a *Strong Prior Knowledge condition*, in which SimStudent was given only strong prior knowledge, and a *Weak Prior Knowledge condition*, in which some of the strong operators were replaced with weak operators. Specifically speaking, the strong operators to get a coefficient, to get a name of a variable in a variable term, to get a denominator, and to get a numerator were omitted. Instead, SimStudent was given weak operators such as to get a first number, to get a first number with sign, and to get a first alphabet letter.

There were also twelve *student* conditions to control a difference in the test problems. In each student condition, there were 13 to 20 training problems. Those training problems were randomly extracted from the same dataset used to analyze student errors in the previous section.

To measure learning gain, the production rules learned by SimStudent were tested on the 11 test problems each time a tutoring was done on a single training problem. A set of 11 test problems were also selected from the same dataset from which the training problems were extracted, but they were semi-randomly selected so that four of the most commonly observed error schemata shown in Table 2 were included.

Notice that since the test problems were extracted from an in vivo study, they contain steps performed by human students. Some of those steps were correct and some were incorrect. To assess the accuracy of the model, we asked SimStudent to predict what step could have made for each intermediate state recorded in a test problem. Namely, we gave SimStudent intermediate states in a test problem one at a time and asked SimStudent which rules could be applied – using a terminology in a literature of production system,

Table 2: A list of the four most commonly observed error schemata appeared in the 11 test problems. In the Error and Problem Schemata, the letters A, B, and C shows a number whereas the letter *v* shows a variable.

Error Schema	Problem Schema
add A	-A = B+C*v*, A-B*v* = C, -A*v* + B = C
subtract A	-A + B*v* = -C, A*v* = B, A = -B*v* – C
multiply A	-A*v* = B, A/*v* = B, A*v* = B
divide A	-A*v* = -B, -A*v* + B = -C, *v*/A = -B

SimStudent computed a *conflict set* for each state the given test problem. We then used an existing Carnegie Learning Algebra I Tutor to evaluate the correctness of individual rule applications in a conflict set.

In each of the 12 student conditions, SimStudent was trained on 113 steps in average (the number of actual training problems varies). Test problems have 140 correct and 28 incorrect steps. For the current study, we only analyzed skills for addition, subtraction, division, and multiplication.

Measurements

To measure the learning outcome, we have conducted both qualitative and quantitative assessment for the production rules learned.

For a quantitative assessment, we computed a *step score* for each step in the test problems as follows: 0 if there is no correct rule application made, otherwise it is a ratio of the number of correct rule applications to the number of all rule applications allowing SimStudent to show all possible rule applications on the step.

For a qualitative assessment, we are particularly interested in errors made by applying learned rules as well as the accuracy of prediction. Given a step *S* performed by a human student at an intermediate state *N*, SimStudent is asked to compute a conflict set on *N*. Rule application R_i ($i = 1, \ldots, n$) is coded as follows:

True Positive: R_i yields the same step as *S*, and *S* is a correct step.

False Positive: R_i yields a correct step that is not same as *S* (*S* may be incorrect).

False Negative: R_i yields an incorrect step that is not same as *S* (*S* may be a correct step).

True Negative: R_i yields the same step as *S* and *S* is an incorrect step.

Results

Impact of Prior Knowledge on Learning

Both the Weak Prior Knowledge (Weak-PK) and Strong Prior Knowledge (Strong-PK) conditions learned skills and the performance on test problems improved as learning proceeded. Figure 1 shows average step score, aggregated across the test problems and student conditions. The X-axis shows the number of training iterations.

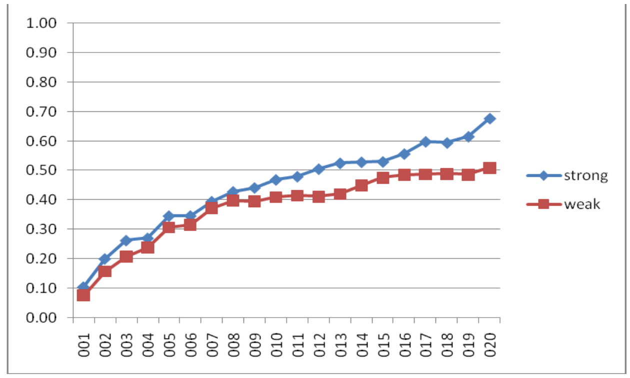

Figure 1: Average step score after each of the 20 training problems for SimStudents with either strong or weak prior knowledge.

The Weak-PK and Strong-PK conditions had similar success rates on test problems after the first 8 training problems. After that, the performance of the two conditions began to diverge. On the final test after 20 training problems, the Strong-PK condition was 82% correct while the Weak-PK was 66%, a large and statistically significant difference (t = 4.00, p < .001). Further, we fit simple power law functions to the learning curves (converting success rate to log-odds) and observed that the slope (or rate) of the Weak-PK learning curve (.78) is smaller (or slower) than that of the Strong-PK learning curve (.82). To test whether this learning rate difference is significant, we subtracted the two functions in their log-log form and verified in a linear regression analysis that the coefficient of the number of training problems (which predicts the difference in rate) is significantly greater than 0 (p < .05).

While it is obvious that differences in prior knowledge can yield to differences in initial performance (as might be measured by a pre-test, this demonstration shows *how differences prior knowledge can also affect the rate at which learning occurs*.

Impact on Prior Knowledge on Error Prediction

Figure 2 shows a number of true negative predictions made on the test problems for each of the training iterations. Surprisingly, the Weak PK condition did make as many as 22 human-like errors on the 11 test problems. On the other hand, the Strong PK condition hardly made human-like errors.

To understand how well SimStudent predicted human-like errors, we computed an accuracy of error prediction, called Error Prediction score, as True Negative / (True Negative + False Negative) on incorrect steps in test problems. Figure 3 shows the average of Error Prediction score for each of the training iterations.

As can be seen in the figure, the Error Prediction score improved for the Weak PK condition as learning proceeded. This implies that SimStudent made more human-like errors than non-human like errors when trained on more problems. This observation further implies that *the proposed model*

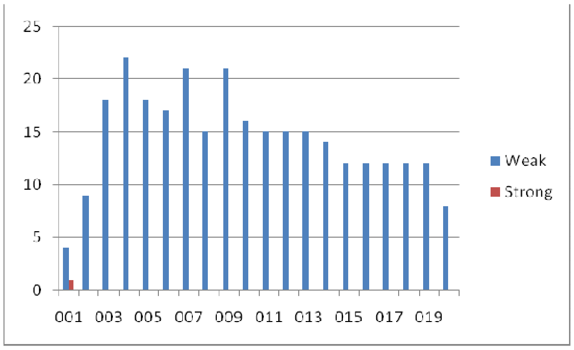

Figure 2: Number of True Negative predictions, which are the same errors made both by SimStudent and human students on the same step in the test problems.

predicts that it is difficult to get rid of human-like errors when the learner does not have Strong prior knowledge.

Table 3 shows the types of human-like errors made by SimStudent and the corresponding type of equations on which the error was made on the test problems.

Although that SimStudent with Weak PK did actually make many human-like errors is an encouraging result, knowing the contents of production rules that SimStudent learned (which reveals the cause of the errors) provides us more knowledge about the impact of Weak PK on learning. The next section shows qualitative analysis of production rules learned with the Weak prior knowledge.

It is, by the way, also worth describing the human-like error made by the Strong PK condition shown in Figure 2. The error was to "add C" to "A*x*+B=-C." It appeared that SimStudent learned this incorrect production rule after observing to "add 3" to the both sides of -3+4*x*=-3, where SimStudent recognized "3" as to be obtained by reversing the sign of the number on the right-hand side. This implies that *human students may also learn incorrect rules from the examples with this kind of ambiguities*.

Production Rules Learned

Recall that we gave the Weak PK conditions three weak operators – first-number, first-number-with-sign, and first-

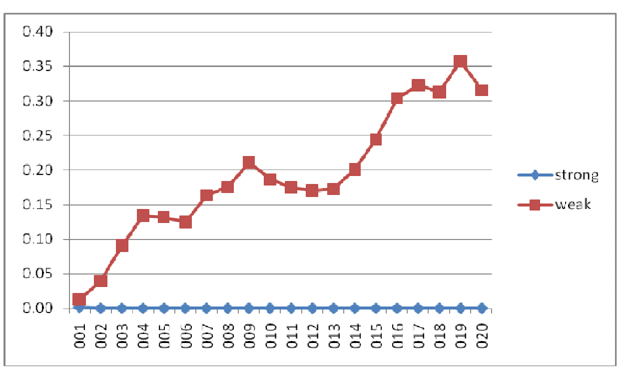

Figure 3: Average of Error Prediction score after each of the 20 training problems for SimStudents with either strong or weak prior knowledge.

Table 3: Errors and problem schemata that appeared during the test as shown in Figure 2.

Error Schema	Problem Schema	Freqency
add B	A = B + C*v*	55
add A	-A*v*+B = C	52
add A	A-B*v* = C	44
add C	A*v*+B = C	23
add C	A*v*+B = -C	23
add A	-A = B+C*v*	22
subtract A	-A+B*v* = C	20
subtract A	-A*v*+B = C	20
divide A	*v*/A = B	14
multiply A	A/*v* = B	11
multiply A	A*v* = B	2
subtract C	A*v*+B = -C	1
subtract A	A = B*v*+C	1

alphabet. All human-like errors shown in Table 3 can be explained using those operators. For example, an error to "add B" for "A = B+C*v*" can be learned as the follows:

IF	right-hand side (RHS) is polynomial
THEN	*get a first number* from RHS, and add that number to both sides

The italicized operation corresponds to a weak operator of first-number. This rule might have learned from A=-B+C*v* gets "add B."

Probably the most striking finding is that *SimStudent sometimes learned correct production rules by combining weak operators*.

In one student condition, SimStudent first learned skill to divide as "when the left-hand side (LHS) has a coefficient and RHS is a constant number then divide both sides by the first number with sign in LHS," which is represented as a production rule as follows:

IF	LHS has a number before alphabet, and RHS is constant number
THEN	*get a first number with its sign* from LHS, and divide both sides with it

This production rule generated a human-like error to "divide A" for "*v*/A=B" during tutoring. SimStudent then revised the rule as follows:

IF	LHS consists of a number and an alphabet
THEN	*get the first alphabet* from the LHS, and compute a quotient of LHS divided by the alphabet, and divide both sides with the quotient

The first two operations in the action part of this production rule are basically extracting a coefficient of a variable term. Namely, SimStudent eventually learned how to take a coefficient of a variable term by combining given weak prior knowledge.

Discussion

In this paper, we showed that SimStudent can be treated as a computational model of human learning, and demonstrated the ability to model the error-prone process of student learning in a complex problem-solving domain. The fundamental hypothesis is that when students rely on more perceptually grounded, shallow prior knowledge then they are more likely to learn incorrect skills.

We have seen the impact of Weak prior knowledge on learning in two ways: (1) although SimStudent learns skills with the Weak prior knowledge, the rate of learning slows down and the accuracy of learned skills is not as good as the ones learned with the Strong prior knowledge, and (2) the Weak prior knowledge leads SimStudent to learn qualitatively different production rules than the ones learned with the Strong prior knowledge. With the Weak prior knowledge, SimStudent often learned incorrect production rules that produced the same errors the human students made.

In prior comparisons of SimStudent with real student data (Matsuda, Cohen, Sewall, Lacerda, & Koedinger, 2007), we found that SimStudent started off behind real students (perhaps because real students have equation solving experience prior to using the tutor), but then quickly passed them. In other words, in these prior runs of SimStudent, which used only strong prior knowledge, the learning rate was too fast relative to human students. The current weak-PK version of SimStudent is not only producing plausible student errors but is learning at a slower rate that may well better correspond with the learning rate of real students. We will explore such a comparison in future work.

In the study shown in this paper, we controlled prior knowledge only for the *operators* to manipulate algebraic expressions. We also noticed that human students often pay attention only to surface (shallow) *features* of the problems. Such skewed perception on features can be modeled as weak feature predicates for SimStudent. Indeed, we have once observed SimStudent learning an overly specific production rules when a key feature predicate was accidentally excluded from a study. In one study, a feature predicate to recognize that a previous step was to apply CLT (combine like term) was accidentally omitted. SimStudent then learned a skill that must follow CLT to be applied when the previous step was neither subtraction nor addition. An impact of having perceptually grounded weak feature predicates along with the weak operators on learning must be tested in the future studies.

Acknowledgments

The research presented in this paper is supported by the National Science Foundation Award No. REC-0537198. This work was also supported in part by the Pittsburgh Science of Learning Center, which is funded by the National Science Foundation Award No. SBE-0354420.

References

Booth, J. L., & Koedinger, K. R. (2008). Key misconceptions in algebraic problem solving. In B. C. Love, K. McRae & V. M. Sloutsky (Eds.), *Proceedings of the 30th Annual Conference of the Cognitive Science Society* (pp. 571-576). Austin, TX: Cognitive Science Society.

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

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

Cypher, A. (Ed.). (1993). *Watch what I do: Programming by Demonstration*. Cambridge, MA: MIT Press.

Koedinger, K. R., Cunningham, K., Skogsholm, A., & Leber, B. (2008). An open repository and analysis tools for fine-grained, longitudinal learner data. In *Proceedings of the international Conference on Educational Data Mining*.

Langley, P., & Ohlsson, S. (1984). Automated cognitive modeling. In *Proceedings of the Fourth National Conference on Artificial Intelligence* (pp. 193-197). Melon Park, CA: AAAI.

Matsuda, N., Cohen, W. W., Sewall, J., Lacerda, G., & Koedinger, K. R. (2007). Predicting Students Performance with SimStudent that Learns Cognitive Skills from Observation. In R. Luckin, K. R. Koedinger & J. Greer (Eds.), *Artificial Intelligence in Education* (pp. 467-476). Amsterdam, Netherlands: IOS Press.

Muggleton, S. (1991). Inductive Logic Programming *New Generation Computing, 8*(4), 295-318.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Sleeman, D. H., Kelly, A. E., Martinak, R., Ward, R. D., & Moore, J. L. (1989). Studies of Diagnosis and Remediation with High School Algebra Students. *Cognitive Science, 13*(4), 551-568.

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

Weber, G. (1996). Episodic learner modeling. *Cognitive Science, 20*(2), 195-236.

Young, R. M., & O'Shea, T. (1981). Errors in Children's Subtraction. *Cognitive Science, 5*(2), 153 - 177.