General and Efficient Cognitive Model Discovery Using a Simulated Student

Nan Li (nli1@cs.cmu.edu)
Eliane Stampfer (estampfe@cs.cmu.edu)
William W. Cohen (wcohen@cs.cmu.edu)
Kenneth R. Koedinger (koedinger@cs.cmu.edu)
School of Computer Science , Carnegie Mellon University
5000 Forbes Ave., Pittsburgh PA 15213 USA

Abstract

In order to better understand how humans acquire knowledge, one of the essential goals in cognitive science is to build a cognitive model of human learning. Moreover, a cognitive model that better matches student behavior will often yield better instruction in intelligent tutoring systems. However, manual construction of such cognitive models is time consuming, and requires domain expertise. Further, manually-constructed models may still miss distinctions in learning which are important for instruction. Our prior work proposed an approach that finds cognitive models using a state-of-the-art learning agent, SimStudent, and we demonstrated that, for algebra learning, the agent can find a better cognitive model than human experts. To ensure the generality of that proposed approach, we further apply it to three domains: algebra, stoichiometry, and fraction addition. To evaluate the quality of the cognitive models discovered, we measured how well the cognitive models fit to student learning curve data. In two of those domains, SimStudent directly discovers a cognitive model that predicts human student behavior better than the human-generated model. In fraction addition, SimStudent supported discovery of a better cognitive model in combination with another automated cognitive model discovery method.

Keywords: cognitive model, machine learning, simulated student

Introduction

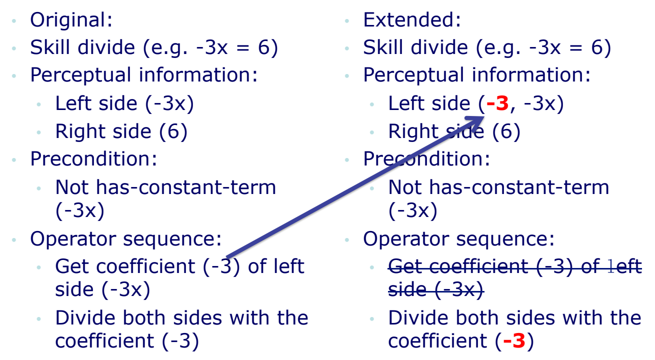
One of the fundamental goals in cognitive science is to understand human knowledge acquisition. A cognitive model of human learning that fits data would be a significant achievement. This goal also complements with another goal in education, which is to provide individualized instruction based on students' abilities, learning styes, etc. Cognitive models provide intelligent tutoring systems with useful information on the learning task difficulties and transfer of learning among similar problems. A better cognitive model often leads to more effective tutoring. A cognitive model is a system that can solve problems in the various ways human students can. One common way of representing a cognitive model is a set of *knowledge components (KC)* (Koedinger & McLaughlin, 2010). The set of KCs includes the component skills, concepts, or percepts that a student must learn to be successful on the target tasks. For example, a KC "divide" in algebra encodes how to proceed given problems of the form $Nv = N$ (e.g., $-3x = 6$), where N stands for a number, and v stands for a variable.

Nevertheless, manual construction of cognitive models remains time consuming and error prone. Traditional ways to construct cognitive models include structured interviews, think-aloud protocols, and rational analysis. Manual construction of cognitive models requires domain expertise, and important instructional details may still be overlooked. Automated search methods such as Learning Factor Analysis (LFA) (Cen, Koedinger, & Junker, 2006) are more objective: the algorithm searches through the space of human-provided factors to find a cognitive model that best matches with human data. Although automated search methods have found better models than manual construction, the quality of the discovered model depends on the quality of the human-provided factors. If there is a better model that can not be expressed by known factors, LFA will not be able to uncover it.

In Li, Matsuda, Cohen, and Koedinger (2011), we have proposed to use the state-of-the-art learning agent, *SimStudent* (Matsuda, Lee, Cohen, & Koedinger, 2009), to automatically discover cognitive models without depending on human-provided factors. SimStudent learns skill knowledge from demonstration and problem solving experience. Each skill SimStudent acquires corresponds to a KC in the cognitive model. To demonstrate the generality of this approach, we present evaluations of the SimStudent-generated models in three domains: algebra, stoichiometry, and fraction addition. We validate the quality of the cognitive models using human student data as in Koedinger and MacLaren (1997). Instead of matching with performance data, we use the discovered cognitive model to predict human learning curve data. Experimental results show that for algebra and stoichiometry, SimStudent directly finds a better cognitive model than humans. For fraction addition, SimStudent results assist LFA in finding a better cognitive model than a domain expert. We have also carried out an in-depth study using Focused Benefits Investigation (FBI) (Koedinger, McLaughlin, & Stamper, 2012) to better understand this machine learning approach, and discussed possible ways of further improvements.

A Brief Review of SimStudent

SimStudent is an intelligent agent that inductively learns skills to solve problems from demonstrated solutions and from problem solving experience. It is a realization of programming by demonstration (Lau & Weld, 1998) using a variation of the version space algorithm (Mitchell, 1982), inductive logic programming (Muggleton & Raedt, 1994), and iterative-deepening depth-first search as underlying learning techniques. For more details, please refer to Matsuda et al. (2009). Recently, in order to build a more human-like intelligent agent, we have developed a model of representation learning, and integrated it into SimStudent's skill acquisition mechanism.



- Original:
- Skill divide (e.g. -3x = 6)
- Perceptual information:
  - Left side (-3x)
  - Right side (6)
- Precondition:
  - Not has-constant-term (-3x)
- Operator sequence:
  - Get coefficient (-3) of left side (-3x)
  - Divide both sides with the coefficient (-3)

- Extended:
- Skill divide (e.g. -3x = 6)
- Perceptual information:
  - Left side (**-3**, -3x)
  - Right side (6)
- Precondition:
  - Not has-constant-term (-3x)
- Operator sequence:
  - ~~Get coefficient (-3) of left side (-3x)~~
  - Divide both sides with the coefficient (**-3**)

Figure 1: Original and extended production rules for divide in a readable format.

Tutoring Strategy

To learn, SimStudent interacts with a tutor (human or automated). Given a problem, if SimStudent does not know how to solve it, it will ask the tutor to demonstrate a next step. If SimStudent knows how to proceed, it will propose the next step, and ask for feedback from the tutor. For each demonstrated step, the tutor specifies a tuple of ⟨*selection, action, input*⟩ (SAI tuple) for a skill along with a *skill label* (e.g., divide). For instance, a demonstrated step for skill "divide" is $\langle(-3x, 6), input\ text, (divide - 3)\rangle$.

SimStudent learns skills as production rules. The left side of Figure 1 shows an example production rule for skill divide in a readable format. A production rule shows "where" to look for useful information (i.e., perceptual information), "when" to apply the skill (i.e., precondition), and "how" to proceed (i.e., operator sequence). To illustrate, consider the rule on the left side of Figure 1. It states that given an equation (e.g., -3x = 6), if the left side does not have a constant term, then first get the coefficient of the left side (-3), and divide both sides by that coefficient.

Each skill corresponds to a KC. During training, the tutor can provide an initial set of KCs to SimStudent by labeling each demonstrated step with a skill name. The label given to SimStudent in the example production rule (i.e., the left side of Figure 1) is "divide". If no such initial KC is known, the tutor can simply label all of the steps with the same skill name. SimStudent's learning algorithm will automatically create the cognitive model as needed.

Skill Learning

SimStudent has three learning components - each acquires one part of the production rules. The first component is a perceptual information (i.e., "where") learner that acquires the path to identify the useful information from its environment. In our case, the environment is a graphical user interface, but it could also be a physical world or an educational game. The elements in the environment are organized in a tree structure. SimStudent learns perception by moving from specific to general. That is, SimStudent tries to find the least general path in the perceptual hierarchy that covers all of the selections in the demonstrated steps. In the example skill "divide," the left and right sides of the equation can be found in the last row that SimStudent entered input.

The second part of the learning mechanism is a precondition (i.e., "when") learner, which acquires the descriptions of desired situations in applying the skill. The learner is given a set of feature predicates to get a basic understanding of the problem. Each predicate is a boolean function that describes relations among objects in the domain (e.g. *(has-coefficient* $-3x$*))*. The precondition learner utilizes FOIL (Quinlan, 1990), an inductive logic programming system that learns Horn clauses from both positive and negative examples. The learning process is general to specific, where the precondition learner starts by considering all situations applicable, and then gradually narrows down the condition based on negative examples. The precondition acquired for the example skill divide is *(not (has-constant ?var-left))*, which returns true if the left side does not have a constant.

The last component is the operator sequence (i.e., "how") learner. The learner is given a set of operator functions as prior knowledge. Operator functions specify (ideally) basic manipulations (e.g. *(add* 1 2*)*, *(get-coefficient* $-3x$*))* that SimStudent can apply to the problem. Given all of the demonstrated steps, the learning mechanism searches for the shortest operator sequence that could explain all of the records, using iterative-deepening depth-first search. For example, given a demonstrated step ⟨*(*$-3x$*,* 6*), (divide* -3*)*⟩, the shortest explanation sequence is *(bind ?coef (get-coefficient ?left-var)) (bind ?output (divide ?coef))*.

There are two groups of operator functions, domain-independent operator functions and domain-specific operator functions. Domain-independent operator functions (e.g. *(add* 1 2*)*) are basic skills applicable across multiple domains. Human students often have knowledge of these simple skills prior to class. Domain-specific operator functions (e.g. *(add-term* $5x - 2$ 5*)*, *(get-coefficient* $-3x$*))* are more complicated skills that human students may not know before class. Thus providing such operators to SimStudent may produce learning behavior that is distinctly different from human students (Matsuda et al., 2009). As we will explain in the next subsection, by integrating representation learning with skill learning, we can reduce or remove SimStudent's dependency on domain-specific operator functions.

Finally, let's talk about how the KCs are discovered. SimStudent starts with a given set of skill labels associated with demonstrated steps. SimStudent tries to learn one rule for each label. It will fail when the perceptual information learner cannot find one path that covers all demonstrated steps, or the operator sequence learner cannot find one operator function sequence that explains all records. In that case, SimStudent learns a disjunctive rule just for the last record. This effectively splits the examples into two clusters. Later, for each new record, SimStudent tries to acquire a rule for each of the clusters with the new record, and stops whenever it successfully learns a rule with one of the clusters. If the new record cannot be added to any of the existing clusters, SimStudent creates another new cluster. By the end of learning, the set of

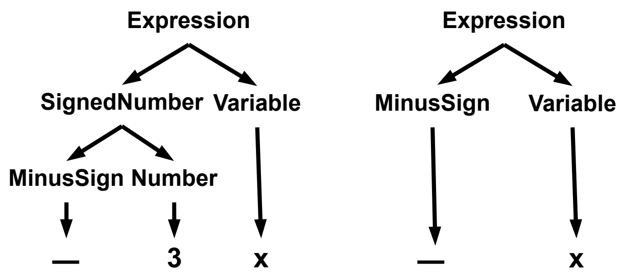


Figure 2: Different parse trees for -3x and -x.

clusters defines a new cognitive model.

Integrating Representation Learning with Skill Learning

As we can see, the prior knowledge given to SimStudent (e.g., the perceptual hierarchy, the operator functions) largely affects the cognitive model it discovers. The more knowledge engineering needed, the less human-like SimStudent is. Therefore, to get a better cognitive model, we need to reduce the amount of knowledge engineering required in constructing SimStudent. Previous studies (Chase & Simon, 1973) have shown that one of the key differences between experts and novices is their different representations of the world. Recently, we have extended SimStudent to support representation learning, and integrated it into skill learning. It has been shown that by integrating representation learning and skill learning, we can automatically learn the tree-structured representation of the problem, and reduce or remove the need of domain-specific operator functions.

The representation learner extends a grammar induction technique to acquire a probabilistic context-free grammar (pCFG) for the problems based on a set of observations (e.g., $-3x$, $2x+5$). To integrate representation learning with skill learning, we extend the perceptual hierarchy to further include the most probable parse trees from the learned pCFG in the contents of the leaf nodes. For example, the left side of Figure 2 is a subtree for parsing "$-3x$" and is connected to the node associated with $-3x$ in the perceptual hierarchy. This subtree ensures that the coefficient -3 is explicitly represented in the perceptual hierarchy. Then, the perceptual information learner and the operator function sequence learner determine how to extract the coefficient from the perceptual hierarchy. This path for identifying the coefficient is added to the perceptual information part of the production rules (See Figure 1, right side). Then, the operator function sequence part no longer needs the domain-specific operator function "get-coefficient". For more details, please refer to Li, Cohen, and Koedinger (2012).

Cognitive Model Discovery Study

In Li et al. (2011), we demonstrated the effectiveness of using SimStudent to discover cognitive models in an algebra domain. In order to evaluate the generality of the proposed approach, in this paper, we tested SimStudent in three domains, algebra, stoichiometry, and fraction addition.

Method

In each domain, we compared the SimStudent model with the best human-generated model available, made by domain experts. To generate the SimStudent model, SimStudent was trained by interacting with automated tutors that simulate the automated tutors used by human students in the studies. The video demonstration in the original study was not used in training SimStudent. SimStudent was trained on problems used by humans students. Then, for each step a human student performed, we assigned the applicable production rule as the KC associated with that step. In cases where there was no applicable production rule, we coded the step using the human-generated cognitive model. Each time a student encounters a step using some KC is considered as an *opportunity* for that student to show mastery of that KC.

To evaluate the quality of the cognitive model, we measured how well the cognitive model fits with human student data using the Additive Factor Model (AFM) (Cen et al., 2006) to validate the coded steps. AFM is an instance of logistic regression that predicts the probability of a student making an error on the next step given each student, each KC, and the KC by opportunity interaction as independent variables.

$$\ln \frac{p_{ij}}{1-p_{ij}} = \theta_i + \sum_k \beta_k Q_{kj} + \sum_k Q_{kj}(\gamma_k N_{ik}).$$

Where:

i represents a student i.

j represents a step j.

k represents a skill or KC k.

p_{ij} is the probability that student i would be correct on step j.

θ_i is the coefficient for proficiency of student i.

β_k is coefficient for difficulty of the skill or KC k

Q_{kj} is the Q-matrix cell for step j using skill k.

γ_k is the coefficient for the learning rate of skill k;

N_{ik} is the number of practice opportunities student i has had on the skill k;

Domains

We carried out our study in three domains: algebra, stoichiometry, and fraction addition. The domains as well as the setup in each domain vary from one to another. This ensures that the experiment tests the generality of the proposed approach.

In algebra, we analyzed data from 71 students who used an Carnegie Learning Algebra I Tutor unit on equation solving. The students were typical students at a vocational-technical school in a rural/suburban area outside of Pittsburgh, PA. A

Table 1: AIC on SimStudent-Generated models and Human-Generated Models.

	Human-Generated Model	SimStudent-Discovered Model
Algebra	6534.07	**6448.1**
Stoichiometry	17380.9	**17218.5**
Fraction Addition	**2112.82**	2202.02

total of 19,683 transactions between the students and the Algebra Tutor were recorded, where each transaction represents an attempt or inquiry made by the student, and the feedback given by the tutor. We selected *40* problems that were used to teach students as the training set for SimStudent.

The stoichiometry dataset contains data from 3 studies. 510 high school and college students participated in the studies, and generated 172,060 transactions. Instructional videos on stoichiometry were intermingled with the problems. Instructional materials were provided via the Internet. It took students from 1.5 hours to 6.5 hours to complete the study. 8 problems in this study were used in training SimStudent.

In fraction addition, we analyzed data from 24 students who used an intelligent tutoring system as part of a larger study. Approximately half of the students were recruited from local schools. Students were given immediate correctness feedback on each step, and were offered on-demand text hints. Each interaction was logged through Datashop, and the 24 students yielded 4558 transactions. SimStudent was tutored with *20* problems from this study.

Measurements

We used Akaike Information Criterion (AIC) and a 10-fold cross validation (CV) to test how well the generated model predicts the correctness of human student behavior. AIC measures the fit to student data and penalizes over-fitting. We did not use BIC (Bayesian Information Criterion) as the fit metric, because based on past analysis across multiple DataShop datasets, it has been shown that AIC is a better predictor of cross validation than BIC is. The cross validation was performed over ten folds with the constraint that each of the training sets must have data points for each student and KC. We reported the root mean-squared error (RMSE) averaged over ten test sets.

Results

As shown in Table 1 and Table 2, in algebra and stoichiometry, the SimStudent-discovered models that have lower AICs and RMSEs ($p < 0.001$) than the human-generated models. This means the SimStudent models better match the data (without over-fitting). However, in fraction addition, the human-generated model performs better than the SimStudent-discovered ones.

A closer look at the models reveals that in algebra, the SimStudent-discovered model splits some of the KCs in the human-generated model into finer grain sizes. For example, SimStudent creates two KCs for division, one for problems of the form $Nv = N$, and one for problems of the form $-v = N$. This is caused by the different parse trees for Nv and $-v$ as shown in Figure 2. Due to this split, the SimStudent-generated model predicts a higher error rate on problems of the form $-v = N$ than problems of the form $Nv = N$. It matches with human student error rates better than the human-generated model, which does not differentiate problems of these two forms.

In stoichiometry, instead of finding splits of existing KCs, SimStudent discovers new KCs that overlap with the original KCs. There are three basic sets of skills in this domain. Within each set, the human-generated KCs are assigned based on the location of the input, while the SimStudent-discovered KCs are associated with the goals of the input. Hence, suppose in two different problems, there are two inputs at the same location in the interface. If they are associated with different goals, the human-generated model will not differentiate them, while the SimStudent-discovered model will put them into two KCs. This indicates that SimStudent not only splits existing KCs, but also discovers totally different KCs.

The fraction addition problem set consists of three types of problems in increasing difficulty: 1) addends have equal denominators; 2) the denominator of one addend is a multiple of the other; 3) addends have unrelated denominators. The human-generated model differentiates these three types of problems in calculating the common denominators and the scaled numerators, and ends up having six KCs. SimStudent, however, associates all of the numerator scaling steps with one KC and associates the common denominator calculations with two KCs. In other words, in this domain, SimStudent partially recovered three out of six KCs, but did not further split them into six KCs. SimStudent did discover the other three KCs, but eventually removed them when they were superseded by more generalized rules. This bias towards more general production rules over specific ones regardless of computational cost appears to be a limitation of SimStudent as a cognitive model. Perhaps if we had let SimStudent keep a utility function for each production rule and retrieve them based on the computational cost, last retrieval time, and correctness, SimStudent may have arrived at all six KCs in the human-generated model.

FBI Analysis and LFA on Fraction Addition

The differences of AIC and RMSE between the models are small. This is partially because the difference between the models is small. FBI, a recently developed technique, is designed to analyze which of these differences improves the model, and by how much. We applied FBI to the SimStudent and human-generated models in each domain to determine

Table 2: CV RMSE on SimStudent-Generated models and Human-Generated Models.

	Human-Generated Model	SimStudent-Discovered Model
Algebra	0.4024	**0.3999**
Stoichiometry	0.3501	**0.3488**
Fraction Addition	**0.3232**	0.3343

why the SimStudent models are better in two of the three cases. In the analysis, we set the human-generated models as the base.

FBI shows that in algebra, splitting "divide" reduces the RMSE of those steps by 1.02%. Further, splitting subtraction and addition decreases the RMSE of those steps by 3.78% and 3.10%, respectively. This also indicates that SimStudent is able to discover KCs of finer grain sizes that match with human data well.

The stoichiometry results are different. SimStudent discovered new KCs that were not part of any existing KCs. Given the 40 KCs in the human-generated model, SimStudent improved 26 of them. The biggest improvement is on skill molecular weight (4.60%), since there are sometimes more than one skill applicable to the same step. The human-generated model misses the additional skill, while the SimStudent model successfully captures both skills.

As described previously, SimStudent did not differentiate the numerator-scaling and common-denominator steps by problem type. This hurts the RMSE of the associated KCs in the SimStudent-generated model. Nevertheless, SimStudent considers finding the common denominator to be a different KC than copying it to the second converted addend. This split decreases by 7.43% for problems with unrelated denominators, and and by 0.12% for denominator steps of problems where one addend denominator was a multiple of the other.

Given the above results, we carried out a third study on fraction addition to test that whether the new KCs created by SimStudent can be used to discover better cognitive models. We used LFA to discover cognitive models given two sets of factors. The baseline LFA model was generated based on the factors (KCs) in the human-generated model. The other LFA model was discovered using both the factors (KCs) in the human-generated model and those in the SimStudent-generated model. Both LFA models were better than the original human-generated model in terms of AIC and RMSE. Moreover, the LFA model using both human-generated and SimStudent-generated factors had better AIC (2061.4) and RMSE (0.3189) than the baseline LFA model (AIC: 2111.96, RMSE 0.3226). In other words, with the help of SimStudent, LFA discovered better models of human students.

Related Work

The objective of this paper is to evaluate the generality of the proposed cognitive model discovery approach. Conati and VanLehn (1999) also applied machine learning techniques to generate cognitive models that fit with human data, but they focused on assessing self-explanation instead of student learning. Additionally, there has been considerable work on comparing the quality of alternative cognitive models. LFA automatically discovers cognitive models, but is limited to the space of the human-provided factors. Other works such as Pavlik, Cen, and Koedinger (2009); Villano (1992) are less dependent on human labeling, but the models generated may be hard to interpret. In contrast, the SimStudent approach has the benefit that the acquired production rules have a precise and usually straightforward interpretation.

Other systems (e.g., Tatsuoka, 1983; Barnes, 2005) use a Q-matrix to find knowledge structure from student response data. Baffes and Mooney (1996) apply theory refinement to the problem of modeling incorrect student behavior. In addition, some research (e.g., Langley & Ohlsson, 1984) uses artificial intelligent techniques to construct models that explain student's behavior in math domains. Besides SimStudent, there has also been considerable research on models of high-level learning (e.g., Laird, Rosenbloom, & Newell, 1986; Anderson, 1993; Taatgen & Lee, 2003; Sun, 2007; Tenenbaum & Griffiths, 2001; Schmid & Kitzelmann, 2011). Other research on creating simulated students (e.g., Chan & Chou, 1997) is also closely related to our work. Nevertheless, none of the above approaches focused on modeling how representation learning affects skill learning. Moreover, none of them compared the system with human learning curve data. To the best of our knowledge, our work is the first combination of the two whereby we use cognitive model evaluation techniques to assess the quality of a simulated learner, and demonstrate it across multiple domains.

Conclusion

In this paper, we evaluated the generality of an automatic cognitive model discovery method, and carried out an in-depth analysis to better understand the proposed approach. To avoid over-generalization of KCs, we would like to further extend the skill learning component to maintain utilities associated with each production rule. Further, we plan to investigate discovery of cognitive models for individual students, to provide more personalized learning. Finally, we plan to further integrate the perceptual learning component into skill learning, so that the representation acquired by the learner is refined during the process learning.

In the study, we show that the integration of the representation learning component into skill learning is key to the success of SimStudent in discovering cognitive models. Results indicate that in two out of three domains, SimStudent-generated models are better predictors of human students' learning performance than human-coded models. For the third domain, when given the SimStudent- and human-generated KCs, LFA finds a better model than the human-generated one. A closer analysis shows that SimStudent is able to split existing KCs into finer grain sizes, discover new KCs, and uncover expert blind spots.

Acknowledgments

The authors would like to thank Hui Cheng for running the experiments. This work was supported by the Pittsburgh Science of Learning Center, NSF Grant #SBE-0836012.

References

Anderson, J. R. (1993). *Rules of the Mind*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Baffes, P., & Mooney, R. (1996). Refinement-based student modeling and automated bug library construction. *Journal of Artificial Intelligence in Education*, *7*(1), 75–116.

Barnes, T. (2005). The Q-matrix method: Mining student response data for knowledge. In *Proceedings aaai workshop educational data mining* (pp. 1–8). Pittsburgh, PA.

Cen, H., Koedinger, K., & Junker, B. (2006). Learning factors analysis - a general method for cognitive model evaluation and improvement. In *Proceedings of the 8th international conference on intelligent tutoring systems* (pp. 164–175).

Chan, T.-W., & Chou, C.-Y. (1997). Exploring the design of computer supports for reciprocal tutoring. *International Journal of Artificial Intelligence in Education*, *8*, 1–29.

Chase, W. G., & Simon, H. A. (1973, January). Perception in chess. *Cognitive Psychology*, *4*(1), 55–81.

Conati, C., & VanLehn, K. (1999). A student model to assess self-explanation while learning from examples. In *Proceedings of the seventh international conference on user modeling* (pp. 303–305). Secaucus, NJ: Springer-Verlag New York, Inc.

Koedinger, K. R., & MacLaren, B. A. (1997). Implicit strategies and errors in an improved model of early algebra problem solving. In *Proceedings of the nineteenth annual conference of the cognitive science society* (p. 382-387). Hillsdale, NJ: Erlbaum.

Koedinger, K. R., & McLaughlin, E. A. (2010). Seeing language learning inside the math: Cognitive analysis yields transfer. In *Proceedings of the 32nd annual conference of the cognitive science society* (pp. 471–476). Austin, TX.

Koedinger, K. R., McLaughlin, E. A., & Stamper, J. C. (2012). Automated student model improvement. In *Proceedings of the 5th international conference on educational data mining* (p. 17-24). Chania, Greece.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). Chunking in soar: The anatomy of a general learning mechanism. *Machine Learning*, *1*, 11–46.

Langley, P., & Ohlsson, S. (1984). Automated cognitive modeling. In *Proceedings of the fourth national conference on artificial intelligence* (p. 193-197). Austin, TX: Morgan Kaufmann.

Lau, T., & Weld, D. S. (1998). Programming by demonstration: An inductive learning formulation. In *Proceedings of the 1999 international conference on intelligence user interfaces* (pp. 145–152).

Li, N., Cohen, W. W., & Koedinger, K. R. (2012). Efficient cross-domain learning of complex skills. In *Proceedings of the eleventh international conference on intelligent tutoring systems* (pp. 493–498). Berlin: Springer-Verlag.

Li, N., Matsuda, N., Cohen, W. W., & Koedinger, K. R. (2011). A machine learning approach for automatic student model discovery. In *Proceedings of the 4th international conference on educational data mining, eindhoven* (p. 31-40).

Matsuda, N., Lee, A., Cohen, W. W., & Koedinger, K. R. (2009). A computational model of how learner errors arise from weak prior knowledge. In *Proceedings of conference of the cognitive science society.*

Mitchell, T. (1982). Generalization as search. *Artificial Intelligence*, *18*(2), 203–226.

Muggleton, S., & Raedt, L. de. (1994). Inductive logic programming: Theory and methods. *Journal of Logic Programming*, *19*, 629–679.

Pavlik, P. I., Cen, H., & Koedinger, K. R. (2009). Learning Factors Transfer Analysis: Using Learning Curve Analysis to Automatically Generate Domain Models. In *Proceedings of 2nd international conference on educational data mining* (pp. 121–130).

Quinlan, J. R. (1990). Learning logical definitions from relations. *Mach. Learn.*, *5*(3), 239–266.

Schmid, U., & Kitzelmann, E. (2011, September). Inductive rule learning on the knowledge level. *Cognitive System Research*, *12*(3-4), 237–248.

Sun, R. (2007, September). Cognitive social simulation incorporating cognitive architectures. *IEEE Intelligent Systems*, *22*(5), 33–39.

Taatgen, N. A., & Lee, F. J. (2003). Production compilation: A simple mechanism to model complex skill acquisition. *Human Factors*, *45*(1), 61–75.

Tatsuoka, K. K. (1983). Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of Educational Measurement*, 345-354.

Tenenbaum, J. B., & Griffiths, T. L. (2001). Generalization, similarity, and bayesian inference. *Behavioral and Brain Sciences*, *24*, 629–640.

Villano, M. (1992). Probabilistic student models: Bayesian belief networks and knowledge space theory. In *Proceedings of the 2nd international conference on intelligent tutoring systems* (p. 491-498). Heidelberg.