In Proceeding of the International Conference on User Modeling, Corfu, Greece, 2007 (to appear)
This is a draft-version - please do not quote

Evaluating a Simulated Student using Real Students Data for Training and Testing

Noboru Matsuda[1], William W. Cohen[2], Kenneth R. Koedinger[1]

[1]Human-Computer Interaction Institute,
[2]Machine Learning Department,
School of Computer Science, Carnegie Mellon University
5000 Forbes Ave., Pittsburgh PA, 15217
[mazda, wcohen, koedinger]@cs.cmu.edu

Abstract: The Simulated Students are machine-learning agents that learn cognitive skills by demonstration. They were originally developed as a building block for the Cognitive Tutor Authoring Tools (CTAT) so that the authors do not have to build a cognitive model by hand, but instead simply demonstrate solutions for the Simulated Students to automatically generate a cognitive model. The Simulated-Student technology could then be used to model human students' performance as well. To evaluate applicability of the Simulated Students as a tool for modeling real students, we applied the Simulated Students to a genuine learning log gathered from classroom instructions using the Algebra I Cognitive Tutor. Such data can be seen as the human students' "demonstrations" on how to solve problems. The results from the empirical study show that the Simulated Students can indeed model human students' performances. After training on 20 problems solved by a set of human students, a cognitive model generated by Simulated Students explained 82% of the problem-solving steps performed correctly by another set of human students.

1 Introduction

Modeling students' cognitive skills is one of the most essential research issues in building Cognitive Tutors, a.k.a. Intelligent Tutoring Systems (Greer & McCalla, 1994). Such a model, called a *cognitive model*, is used, for example, to assess students' performance and to provide feedback (model tracing), to monitor progress in students' learning over the course of problem-solving to plan instructional strategies adaptively (knowledge tracing), or simply to give a hint on what to do next (Anderson *et al.*, 1995). Yet, developing a cognitive model is a labor-intensive task that forces even a skilled expert to work for hundreds of hours.

We have developed a machine learning agent – called a *Simulated Student* – that learns cognitive skills from demonstration. The Simulated Student is designed to be used as a building block of intelligent authoring tools for Cognitive Tutors, called the Cognitive Tutor Authoring Tools, or CTAT (Aleven *et al.*, 2006). Using this technology, an author can simply demonstrate a few solutions. The Simulated Student generalizes those solutions and generates a cognitive model that is sufficient to explain the solutions. Such a cognitive model would then be embedded into the Cognitive Tutor as the knowledge base for model-tracing and, this way, relieve authors from the burden of building a cognitive model by hand.

The goal of the Simulated Student project is twofold: on the engineering side, we investigate whether the Simulated Students could facilitate the authoring of the Cognitive Tutors. On the user modeling side, we explore whether the Simulated Students could help us advance studies in human and machine learning.

As a step towards the first goal, we have tested the Simulated Students on several domains including algebra equation, long division, multi-column multiplication, fraction addition, Stoichiometry (chemistry), and Tic-Tac-Toe. So far, the Simulated Students showed a reasonable and stable performance on those test domains (Matsuda *et al.*, 2006).

The goal of this paper, as an attempt to address the second goal mentioned above, is to see whether the Simulated Students could model cognitive skills acquired by human students during learning by solving problems. To address this issue, we have applied Simulated Students to the student-tutor interaction log data (i.e., the record of activities colleted while human students were learning with a computer tutor) to see whether the Simulated Students could learn the same cognitive skills that the human students learned. In other words, we considered the human students' learning log as the "demonstrations" performed by individual human students. We fed the Simulated Students with those demonstrations and had them learn cognitive skills. If the Simulated Students could indeed learn cognitive skills in this way, then we would be able to use Simulated Students to investigate human students' learning by analyzing cognitive models generated by the Simulated Students as well as its learning processes.

The fundamental technology that supports Simulated Students is inductive logic programming (Muggleton & de Raedt, 1994) and programming by demonstration (Lau & Weld, 1998). There are studies on using machine-learning agents for cognitive modeling and educational tools, for example, (Baffes & Mooney, 1996; Johnson *et al.*, 1998; Mertz, 1997). Probably the most distinctive aspect of the Simulated Student developed for the current study is that it generates human readable (hence editable) production rules that model cognitive skills performed by humans.

The outline of the paper is as follows. We first introduce the Cognitive Tutor that the human students used in the classroom. This gives a flavor of how human students "demonstrated" their skills to the Cognitive Tutor. We then explain how the Simulated Students learn cognitive skills from such demonstrations. Finally, we show results from an evaluation study on the applicability of the Simulated Students to the genuine student-tutor interaction log data.

2 Algebra I Cognitive Tutor

The Algebra I Tutor is a Cognitive Tutor developed by Carnegie Learning Inc. This tutor is used in real classroom situations for high school algebra at about 2000 schools nationwide in the United States (Koedinger & Corbett, 2006). For the current study, we use human students' log data collected from a study conducted in a high school in an urban area of Pittsburgh. There were 81 students involved in the study. The students used the Cognitive Tutor individually to learn algebra equation solving. There were 15 sections taught by the tutor, which covered the most of the skills necessary to solve linear equations. In this paper, we only use the log data collected through the first four sections. In those introductory sections, the equation only contains one unknown and the form of equation is A+B=C+D where A, B, C, and D are either a constant or an unknown term in the form of N*x* or *x*/N where N is a number.

2.1 Problem solving with Algebra I Tutor

The tutor logged the students' activities in great detail. For the current study, however, we only focus on the *problem-solving steps*, which are slightly different from *equation-transformation steps*. Explanations follow.

There are two types of problem-solving steps: (1) an *action step* is to select an algebraic operation to transform an equation into another (e.g., "to declare to add $3x$ to the both sides of the equation"), and (2) a *type-in step* is to do a real arithmetic calculation (e.g., "to enter –4 as a result of adding $3x$ to $-4-3x$"). Performing these problem-solving steps, a given equation is transformed as follows: a student first selects an action and then applies it to both sides of the equation. For example, for an equation shown in **Fig. 1** (a), the student first selected "Add to both sides" from the pull down menu (b), which in turn prompts the student to specify a value to add (c). This completes the first problem-solving step, which by definition is an action step. The student then enters the left- and right-hand sides separately. The **Fig. 1** (d) shows a moment where the student had just typed-in the left-hand side. Thus, entering a new equation is completed in two problem-solving steps, which are both type-in steps. In sum, three problem-solving steps correspond to a single equation-transformation step that transforms an equation into another. Sometimes, however, the tutor carries out the type-in steps for the student, especially when new skills have just been introduced.

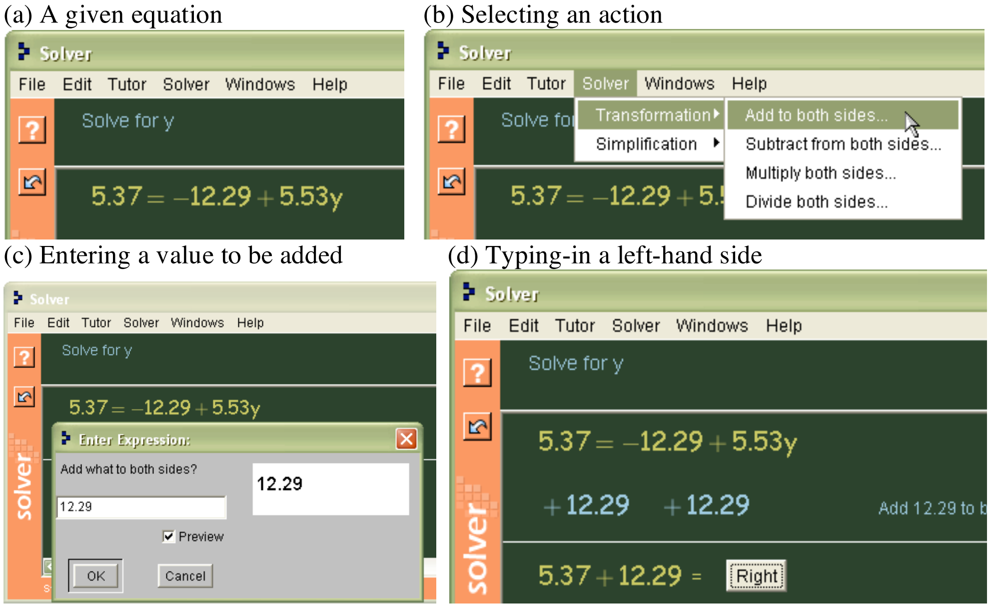

Fig. 1. Screen shot from the Algebra I tutor

When a student makes an error, the tutor provides feedback. The student can also ask for a hint (by pressing the [?] button on the left side of the tutor window) when he/she gets stuck.

Every time a student performs a step, the tutor logs it. The log contains, among other things, (1) the equation in which the step was made, (2) the action taken (either the name of the algebraic operation selected from the menu for an action step, or the symbol "type-in" for a type-in step), (3) the value entered (e.g., the value specified to be added to the both sides for the "add" action mentioned above, or the left- and right-hand side entered for the type-in steps), and (4) the "correctness" of the step, which is either "correct" (in case the student's steps is correct) "error" (the student's steps is incorrect), or "hint" (when the student asked a hint).

2.2 Model tracing and cognitive modeling

As mentioned above, when a student performs a problem-solving step, the tutor provides immediate feedback on it. This is possible because the tutor has a *model* (a *cognitive model*) of the target cognitive skills, represented as a set of production rules. Since a cognitive model usually contains production rules not only for correct steps, but also for incorrect steps, the tutor can provide feedback on typical errors.

Although model tracing is a powerful technique that plays the most essential role in the Cognitive Tutors, building a cognitive model is a labor intensive task. This gave us a reason to develop a Simulated Student and integrate it into the Cognitive Tutor Authoring Tools. The Simulated Students learn cognitive skills by observing problem-solving steps demonstrated by human experts. The next section provides an overview of this issue.

3 Overview of Simulated Students

This section is a brief overview of the Simulated Students. We first explain how the Simulated Students learn cognitive skills from demonstration. A double meaning of "demonstration" in the current context would be then explained – a demonstration by an author who is building a Cognitive Tutor, and a "demonstration" in a learning log made by human students. We then explain briefly how the Simulated Students learn a cognitive model. Due to the space limitation, we do not provide details of the learning algorithms. See (Matsuda *et al.*, 2005) for more details.

3.1 Cognitive modeling with Simulated Students

For the sake of explanation, **Fig. 1** shows a sample student interface for a Cognitive Tutor to teach algebra equation solving. In this particular tutor, an equation is represented in a simple table. An equation (e.g., "41.72y+87=34.57") is transformed into another equation (e.g., "41.72y=34.57-87") with three *problem-solving steps*: (1) specify an action, e.g., "subtract 87 (from the both sides)", (2) enter LHS, e.g., "41.72y", and (3) enter RHS, e.g., "34.57-87."

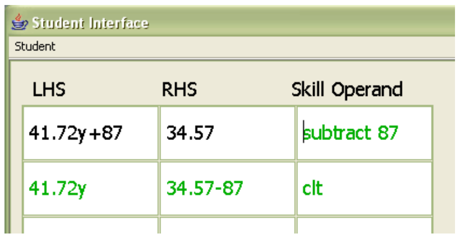

Fig. 1. A sample tutor interface for algebra equation solving.

A Simulated Student learns a single production rule for each of the problem-solving steps demonstrated in the student interface. When demonstrating a problem-solving step, the demonstrator must specify two things; (1) a focus of attention, and (2) a skill name. The *focus of attention* are the elements on the student interface from which the value to be entered is determined. For example, in **Fig. 1**, the first problem-solving step ("subtract 87") requires two elements, "41.72y+87" and "34.57," as the focus of attention. The *skill name* must be unique for unique steps and consistent throughout the demonstration. In the above example, a skill to enter "subtract 87" is called "subtract," and a skill to enter "41.72y" and "34.57-87" are both called "subtract-typein." The actual value entered (e.g., "subtract 87") is called an "input."

The tutor interface shown in **Fig. 1** is also used in the current study as an interface for the Simulated Students to model human students' performances. It is a simple but straightforward realization of the human students' performances in a Simulated-Student readable form. There is an issues on focus of attention to be mentioned here. When the human students were using the Algebra I Tutor, they did not indicate their focus of attention, and hence no information of focus of attention is stored in the log. We have presumed that both LHS and RHS are used as the focus of attention for the action steps. Likewise, for the type-in steps, we presume that the Skill Operand and the cell immediately above the cell to be typed-in are the focus of attention. So, for example, if "34.57-87" is entered, which is a skill "subtract-typein", the elements "34.57" and "subtract 87" are used as the focus of attention.

3.2 Learning algorithm

Production rules are represented in the Jess production rules description language (Friedman-Hill, 2003). A production rule used in the Cognitive Tutors consists of three major parts: (1) WME-paths, (2) feature conditions, and (3) an operator sequence. The first two components construct the left-hand side of a production rule, which specifies which elements of the interface are involved in the production rule, and what conditions should hold about those elements in order for the production rule to be fired. The operator sequence constitutes the right-hand side actions of the production rule, which specifies what should be done with the interface elements to make the "input" value of the step (see the definition in 3.1).

The Simulated Student utilizes three different learning algorithms to learn three components (the WME-path, the feature conditions, and the operator sequence) separately. An example would best explain how. Suppose a step is demonstrated and named as N. Also suppose that this is the k-th *instance* of demonstration for the skill N. Let's denote this as $I(N,k)$. Let's assume that the skill N requires two elements as focus of attention, and we denote them as $<F^{N,k}_1, F^{N,k}_2>$, the elements of focus of attention for the k-th instance of the skill N.

WME-path is a straightforward generalization of the focus of attention. The elements specified in focus of attention are elements on the tutor interface. They can thus be uniquely identified in terms of their "location" in the interface. Suppose, for example, the first element of focus of attention in the j-th instance of the skill N, $F^{N,j}_1$, is "a cell in the 1st column on the 2nd row." If the first element of focus of attention in the (j+1)-th instance $F^{N,j+1}_1$ is "a cell in the 1st column on the 3rd row," then the WME-path for the 1st element of focus of attention for the skill N would be "a cell in the 1st column at any row."

Simulated Students use FOIL (Quinlan, 1990) to learn feature conditions. The target concept is an "applicability" of the skill N given the focus of attention $<F^N_1, F^N_2>$, or in a prolog-like form $N(F^N_1, F^N_2)$. When a step $I(N,k)$ is demonstrated, it serves as a positive example for the skill N, and a negative example for all other skills. Basically, as the demonstration proceeds, the skill N has all $<F^{N,k}_1, F^{N,k}_2>$ as positive examples, and $<F^{X,k}_1, F^{X,k}_2>$ as negative examples, where X is all the other skills demonstrated. We provide FOIL with a set of *feature predicates* as the background knowledge to compose hypotheses for the target concept. Some examples of such feature predicates are isPolynomial(A), isNumeratorOf(A,B), isConstant(A). Once a hypothesis is found for the target concept, the body of the hypothesis becomes the feature condition in the left-hand side of the production rule. Suppose, for example, that FOIL found a hypothesis $N(F^N_1, F^N_2)$:- isPolynomial(F^N_1), isConstant(F^N_2). The left-hand feature condition for this production rule would then say that "the value of the first focus of attention must be polynomial and the second value must be a constant."

The Simulated Students apply iterative-deepening depth-first search to learn an operator sequence for the right-hand side of the production rules. When a new instance of demonstration on skill N is provided, the Simulated Students search for a shortest operator sequence that derives the "input" from the focus of attention for the all instances demonstrated. Those *operators* are provides prior to learning as background knowledge.

4 Evaluation

To evaluate the applicability of the Simulated Students technology to genuine real students' learning log, we have conducted an evaluation study to see (1) whether the Simulated Students can generate cognitive models for the real students' performances, and if so (2) how accurate such models are.

4.1 Data

The students' learning log was converted into *problem files* that Simulated Student can read. Each problem file contains the sequence of problem-solving steps made by a single student to solve a single problem. There were 13451 problem-solving steps performed by 81 human students. These problem-solving steps were converted into 989 problems.

4.2 Method

We applied the validation technique. The 81 students were randomly split into 14 groups. Each of those 14 groups were used exactly once for training and once for testing. More precisely, for the *n*-th validation, the *n*-th group is used for training and the (*n*+1)-th group is used for testing. A total of 14 validation sessions were then run.

During training, a Simulated Student learned cognitive skills only on those steps that were correctly performed by the human students. In other words, Simulated Students learned only the correct skill applications "demonstrated" by the human students.

Because of memory limitations, we could use only as many as 20 training and 30 test problems in each of the validation sessions. To select those problems, a human student was randomly selected in a given group. If the selected human student did not solve enough problems, then more human students were selected randomly. In total, 280 training and the 420 test problems were used across the 14 validation sessions.

In a validation session, the 30 test problems were tested *each time* after a Simulated Student was trained on one training problem. Since there were 20 training problems, the total of 600 attempts of model-tracing a test problem were made during each validation session. There were 32 operators and 12 feature predicates used as the background knowledge.

4.3 Results

In two out of 14 validation sessions, we identified corrupted data and could not complete runs on these. In one validation session, not all cognitive skills discussed below appeared in the training problems. Hence there are 11 validation sessions (220 training and 330 test problems) used for the analysis discussed in the rest of the section.

4.3.1 Learning opportunities

There were 12 skills involved in the training problems. Eight of them are action skills and another four are type-in skills. Four out of the eight action skills were learned in only a very few training problems and they did not appear in all validation sessions. Therefore, we have excluded those skills from the analysis. In sum, there were four action skills and four type-in skills included in the current analysis. **Table 1** shows frequency of learning for each of those skills. The skills add, subtract, multiply, and divide are action skills. The skill "add," for example, is to add a term to both sides. The skill "add-typein" is for a type-in step that follows the step "add." Note that those eight skills are the most basic skills to solve simple equations.

Table 1. Frequency of learning for each skill appearing in the training problems. The numebrs on the first row are the ID's for the validaion sessions. The validation sessions and the skills are sorted on the total number.

Skill	014	010	009	004	008	011	006	003	001	005	007	Total	Ave.
divide	22	21	22	20	22	19	20	21	20	21	20	228	20.73
divide-typein	20	16	18	18	14	12	12	10	10	10	12	152	13.82
subtract	15	18	12	14	13	11	16	9	6	11	7	132	12.00
add	7	4	10	6	10	8	5	12	14	10	13	99	9.00
subtract-typein	14	16	8	12	10	6	10	4	2	4	6	92	8.36
multiply	9	10	9	6	8	11	9	10	8	6	6	92	8.36
add-typein	6	2	10	6	8	6	2	6	8	6	6	66	6.00
multiply-typein	6	8	4	6	2	6	4	4	6	6	2	54	4.91
Total	99	95	93	88	87	79	78	76	74	74	72	915	83.18

4.3.2 Learning curve analysis

To analyze how Simulated Students' learning improved over the time, we measured the "accuracy" of production rules on the test problems. Each time learning is completed on a training problem, each of the steps in the 30 test problems were model-traced using the production rules available at that moment. An attempt of model-tracing is defined to be "successful" when there is a production rule with the LHS conditions that hold and the RHS operator sequence generates an "input" that matches the step. **Fig. 2** shows the learning curves aggregated across all eight skills and averaged across the 11 validation sessions. **Fig. 3** shows the learning curve for the individual skills.

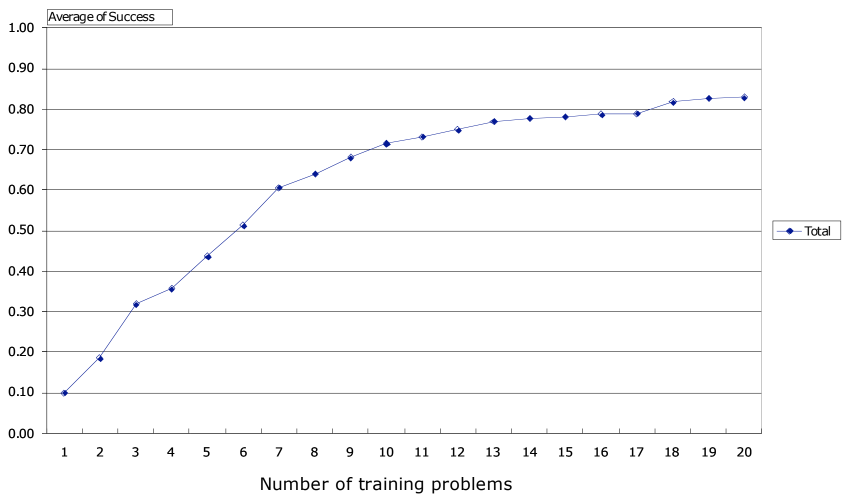

Fig. 2. Overall performance improvement in terms of the average ratio of successful model-tracing aggregated across all validation sessions and the (eight) skills. The x-axis shows the number of training problems.

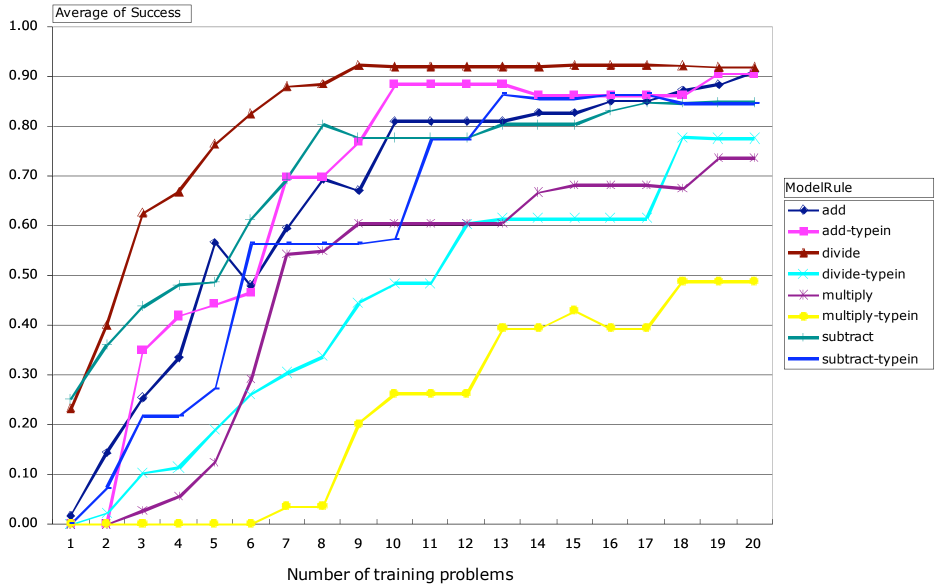

Fig. 3. Learning curve on individual skills. The learning curve shown in **Fig. 2** is decomposed into individual skills.

Overall, the Simulated Students learned skills quite well. After trained on 20 problems, the ratio of model tracing reached at least 73% on most of the skills. However, some skills were not exactly learned as well as the other skills – it seems to be difficult to learn the skill "multiply-typein." It turned out that not all skills had the same number of opportunities to be learned. Different training problems have different solution steps hence contain different number of instances for each of the skills to be demonstrated.

Fig. 4 shows how the accuracy of model tracing grew as the Simulated Students had more and more opportunities to learn individual skills. The x-axis shows a *frequency of learning* (in contract to the number of problems demonstrated). The y-axis shows the *average of the average ratio* of successful model-tracing aggregated from the beginning when a certain number of learning occurred. This graph shows how quickly (or slowly) the learning occurred. For example, even when two skilled ended up with having the same performance rate (e.g., the skills "add" and "add-typein" shown in **Fig. 3**), it can be read from **Fig. 4** that the skill "add-type" reached the final performance quickly only within 5 instances of demonstration.

The four action skills, add, subtract, multiply, and divide, were learned in the same speed. Different type-in skills had different speed and the quality of the production rules (i.e., the accuracy of model-tracing) differed significantly. We have yet to investigate a reason of the difference in the learning speed on these skills.

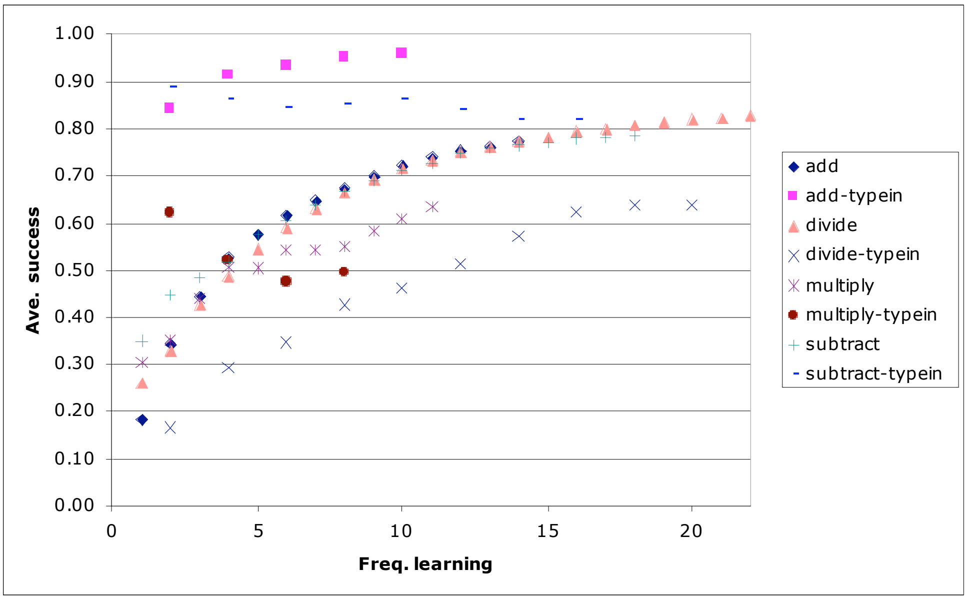

Fig. 4. Average of the average ratio of successful model-tracing in the first *x* opportunities of learning. For example, for the skill "add," the average successful ratio for the first 3 learning opportunities were .18, .50, and .64. Therefore, on the above graph, the value for the 3rd plot for add is .44.

5 Conclusion

We have showed that the Simulated Students can indeed model human students' performances from their learning activity log. The accuracy of model-tracing based on the cognitive model generated by the Simulated Students reached 83% after trained on 20 problems performed by human students.

As long as the human students exhibit correct performances (i.e., the performances are consistent), even when they have variants in strategy and representations, the Simulated Students can generate a cognitive model that is consistent with the human students' (correct) performances. We have yet to improve the learning ability of Simulated Students so that the human students' incorrect behaviors can be modeled. This is one of the important issues to be addressed in the future.

The above finding on the ability of the Simulated Students to model real students' performance suggests potential ways to expand the applicability of Simulated Students. First, if we can model human students' erroneous performances as well, then it might be possible to predict human students' performance on a novel problem. Technically speaking, modeling "incorrect" performances does not differ greatly from modeling a "correct" performance, if the human student makes a systematic error (based on a stable misconception). The real challenge would then be how to deal with the inconsistent behaviors (e.g., guess, slip, or even "gaming").

If the Simulated Students could actually model not only the human students' correct performances but also incorrect ones, then Simulated Students would be used as a building block of the Cognitive Tutor (in contrast to a building block of the authoring tools). Because a Simulated Student would be able to solve a problem with a particular human student's cognitive model, it would be possible to propose a problem(s) that the human student might fail to solve. This idea of using interpretable student models is old, but by using Simulated Students, we might be able to propose a domain-independent framework for this kind of student modeling.

Reference:

Aleven, V., McLaren, B. M., Sewall, J., & Koedinger, K. R. (2006). The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains. In M. Ikeda, K. D. Ashley & T. W. Chan (Eds.), *Proceedings of the 8th International Conference on Intelligent Tutoring Systems* (pp. 61-70). Berlin: Springer Verlag.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences, 4*(2), 167-207.

Baffes, P., & Mooney, R. (1996). Refinement-Based Student Modeling and Automated Bug Library Construction. *Journal of Artificial Intelligence in Education, 7*(1), 75-116.

Friedman-Hill, E. (2003). *Jess in Action: Java Rule-based Systems*. Greenwich, CT: Manning.

Greer, J. E., & McCalla, G. (1994). *Student modelling: the key to individualized knowledge-based instruction*. Berlin; New York: Springer-Verlag.

Johnson, W. L., Rickel, J., Stiles, R., & Munro, A. (1998). Integrating pedagogical agents into virtual environments. *Presence, 7*(6), 523-546.

Koedinger, K. R., & Corbett, A. (2006). Cognitive Tutors: Technology Bringing Learning Sciences to the Classroom. In R. K. Sawyer (Ed.), *The Cambridge Handbook of the Learning Sciences* (pp. 61-78). New York, NY: Cambridge University Press.

Lau, T. A., & Weld, D. S. (1998). Programming by demonstration: an inductive learning formulation. In *Proceedings of the 4th international conference on Intelligent user interfaces* (pp. 145-152). New York, NY: ACM Press.

Matsuda, N., Cohen, W. W., & Koedinger, K. R. (2005). Applying Programming by Demonstration in an Intelligent Authoring Tool for Cognitive Tutors. In *AAAI Workshop on Human Comprehensible Machine Learning (Technical Report WS-05-04)* (pp. 1-8). Menlo Park, CA: AAAI association.

Matsuda, N., Cohen, W. W., Sewall, J., & Koedinger, K. R. (2006). Applying Machine Learning to Cognitive Modeling for Cognitive Tutors. In.

Mertz, J. S. (1997). Using A Simulated Student for Instructional Design. *International Journal of Artificial Intelligence in Education, 8*, 116-141.

Muggleton, S., & de Raedt, L. (1994). Inductive Logic Programming: Theory and methods. *Journal of Logic Programming, 19-20*(Supplement 1), 629-679.

Quinlan, J. R. (1990). Learning Logical Definitions from Relations. *Machine Learning, 5*(3), 239-266.