Tuning Cognitive Tutors into a Platform for Learning-by-Teaching with SimStudent Technology

Noboru Matsuda[1], William W. Cohen[1], Kenneth R. Koedinger[1], Gabriel Stylianides[2], Victoria Keiser[1], Rohan Raizada[1]

[1]Carnegie Mellon University
5000 Forbes Ave. Pittsburgh PA, 15213
[2]University of Pittsburgh
5517 Posvar Hall, Pittsburgh PA 15260 USA
[noboru.matsuda, wcohen, koedinger, keiser, rohanr]@cs.cmu.edu
gstylian@pitt.edu

Abstract: To study cognitive and social factors that facilitate the tutor-learning effect, we have developed an on-line game-like environment where students learn algebra equation solving by teaching a computer agent, called SimStudent. SimStudent is a first pedagogical teachable agent that commits to genuine inductive learning *and* studied in authentic classroom settings. Our Learning by Teaching (LBT) environment is also designed to be highly modular and domain independent. Furthermore, the tutoring interface used in the proposed LTB environment is automatically extracted from a Cognitive Tutor authored with Cognitive Tutor Authoring Tools. Thus, it is fairly easy to build a LBT environment for a new subject domain.

Key words: Teachable Agent, Learning by Teaching, SimStudent, Cognitive Tutor, Inductive Logic Programming, Machine Leaning

1 Introduction

It is well known that students learn by teaching others [1], and there is a school of researchers studying such an *effect of tutor learning* using a cutting-edge technology of pedagogical computer agent. The advanced agent technology enables us to conduct fine-grained controlled studies to investigate cognitive and social factors that facilitate tutor learning.

One of the challenging issues to study the effect of tutor learning is that the tutor learns at the tutee's expense. The tutee might not learn much from the tutor who is also learning the subject. It is also difficult to conduct control studies to explore the facilitators for the tutor learning in such a real peer-learning context. To address these issues, we have developed a pedagogical machine-learning agent, called SimStudent [2] that inductively learns cognitive skills from worked-out examples or through tutored problem-solving in the context of learning by teaching, which is the primary focus in the current paper. Using the SimStudent technology, we developed an on-line game-like learning environment where students learn algebra equation solving by teaching SimStudent.

The purpose of this paper is to introduce an overview of SimStudent and the Learning by Teaching (LBT) environment. We discuss a various aspects of the teachable agent and summarize advantages and disadvantages of SimStudent as a research tool to study the effect of tutor learning. One of the unique characteristics of our LBT environment is that it is designed to use a tutoring interface taken from a Cognitive Tutor, which is authored by Cognitive Tutor Authoring Tools (CTAT) [3]. Coupling CTAT and SimStudent strikingly makes it affordable to build a new LBT environment with customized study variables to advance the theory of tutor-learning effect.

2 Teachable Agent Technologies

A *teachable agent* is a peer learner that students can teach. Using a teachable computer agent in an educational context is not a new idea. There have been a number of different teachable agents developed so far for different purposes and hence with different roles.

One of the most controversial issues is whether a teachable agent should actually learn knowledge from students or it could just dissimulate its learning capability. Math Concept Learning System (MCLS) [4] is an early example of the teachable agent that engages in *inductive learning* from examples. Our SimStudent also falls into this category. A set of background knowledge is given to compose the hypotheses from given examples. Since this type of teachable agent has the ability to learn correct or incorrect knowledge based on the student's input, it can be used to see if students learn from errors made by the teachable agent, the so-called effect of corrective self-explanation. The other type of teachable agent does not actually commit to learning, but rather solicits tutoring activities from the student [5]. There has been no direct control study comparing teachable agents that commit genuine learning vs. pseudo learning. Such a comparison would clarify the importance of the behavioral characteristics of the teachable agent for tutor learning.

Sometimes, the domain principles are conveyed directly by the student using the exact knowledge representation used by the teachable agent. Other teachable agents create such domain principles by themselves using their own knowledge representation that may be different from the students' mental models. Betty's Brain [6] is an example of the teachable agent that shares the knowledge representation with the student. When teaching Betty's Brain, the student draws a concept map representing a causal network in a natural system (e.g., ecosystem). Given the concept map, Betty's Brain then derives a causal inference. When Betty's Brain makes an incorrect inference, the student must identify a flaw in the concept map and correct it by redrawing the map. DENISE [7] is another example of sharing a knowledge representation to learn a causal qualitative model of economics. Since MCLS and SimStudent both learn production rules, the student does not exactly know what the teachable agent has learned. Such a gap between the student's input and the teachable agent's output then gives the student more challenge to remediate the incorrect knowledge acquired by the teachable agent. Thus, the formative assessment becomes more natural and essential in the tutoring context. This issue is also related to the visibility of the acquired knowledge discussed in the next section.

Would it facilitate the student's learning if the student could directly peek at the knowledge that the teachable agent has learned? Diagnosing the proficiency of the tutee and providing an adaptive instruction is an essential aspect of tutoring. If the student could directly itemize what the teachable agent knows, it might facilitate the tutoring processes. In some systems, such a direct observation happens quite naturalistically. Obayashi et al. [8] developed a virtual classroom environment where multiple teachable agents, after being tutored by individual students, solve problems. The students observe the answers made by the teachable agents (not only their own agents, but also others), and reflect their own knowledge. The subject domain used for their study was psychophysiology, and hence the student and the teachable agent shared the knowledge representation. Betty's Brain is another example in which the student can directly browse the knowledge acquired by the teachable agent (which in this case is exactly what the student has drawn).

3 SimStudent – General Overview

SimStudent is a machine-learning agent that inductively learns cognitive skills for solving procedural problems from examples. It is a realization of *programming by demonstration* with an underlying technology of *inductive logic programming*. There are two essential learning strategies implemented for SimStudent – learning from worked-out examples and learning by tutored-problem solving. In either case, there must be a *tutor agent* that provides examples and feedback to SimStudent.

When SimStudent is engaged in the former learning strategy, SimStudent attempts to generate a set of hypotheses that explain demonstrated solutions. The hypotheses are represented as production rules. This type of learning is passive and only *positive examples* are explicitly given (in the form of worked-out examples) to SimStudent. A closed world assumption applies here, so a positive example of a particular skill *K* implicitly serves as a *negative example* for all other skills than *K*.

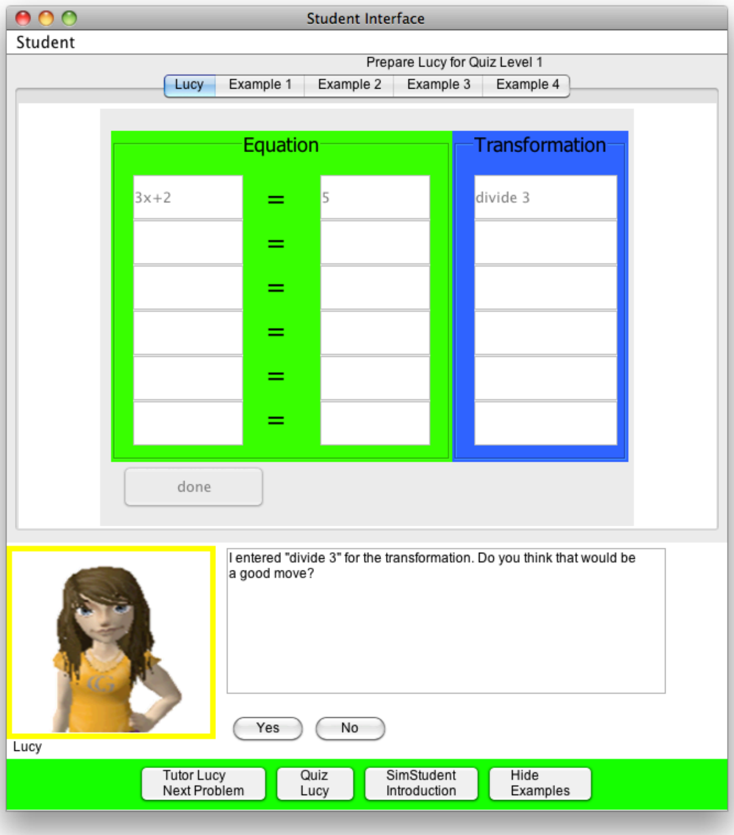
When SimStudent commits learning by tutored-problem solving, it is given a series of problems to solve. While solving problems, SimStudent gets feedback from the tutor agent on the correctness of the step performed. The feedback on the correctness simply shows whether the step performed is correct or not. SimStudent may commit alternative attempts until the step is performed correctly. When SimStudent cannot perform a step correctly, then SimStudent asks the tutor agent for a hint on what to do next. The tutor agent then responds to the request by actually performing the step. Details of SimStudent can be found elsewhere [2].

4 Learning by Teaching Environment

Fig. 1 shows a screenshot of the LBT environment. Although the underlying system architecture is domain independent, the current system is built for algebra linear equations. SimStudent is visualized as an avatar at the lower left corner, and Lucy is the name of the avatar in the current version of the system. The tutor agent in this learning environment obviously is a student who tutors Lucy.

There is a Tutoring Interface in the LBT environment that the student and Lucy share to solve problems. The student enters a problem for Lucy, and Lucy attempts to solve it. A step performed by Lucy is shown in the Tutoring Interface. The student then provides feedback on the correctness of the step performed. In Fig. 1, Lucy divided the equation 3x+2=5 by 3, thus entered "divide 3" in the Transformation cell. Lucy then asked the student if such a move was a good move or not. The student provides feedback by clicking the [Yes]/[No] button. Since the student is learning the equation solving, he/she may provide incorrect feedback. The "correctness" of Lucy's performance is determined merely by the student's feedback.

The goal for the student in this LBT environment is to have Lucy pass the quiz. The system developer prepares the quiz items. When the [Quiz Lucy] button is clicked, Lucy takes the quiz. The results are then summarized in a separate window as shown in Fig. 2.

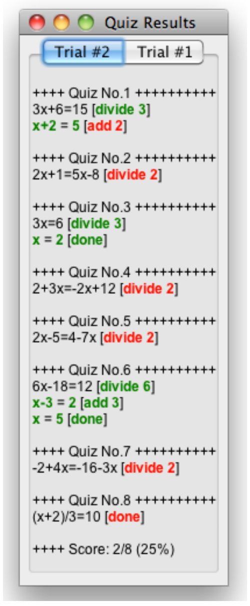


Equation			Transformation
3x+2	=	5	divide 3

I entered "divide 3" for the transformation. Do you think that would be a good move?

Fig. 1. Screenshot of the Learning by Teaching Environment. SimStudent is visualized as an avatar at the lower left corner and names as Lucy.

Since SimStudent is capable of inductive learning, we can control SimStudent's learning ability by manipulating SimStudent's background knowledge and, thus, incorporate in SimStudent specific misconceptions that will yield certain errors when solving problems [9]. We have analyzed errors that students commonly make and successfully trained SimStudent to make the same errors when it is first launched. This gives us the opportunity to examine how students deal with these errors as they come up when they tutor Lucy.



```
++++ Quiz No.1 ++++++++++
3x+6=15 [divide 3]
x+2 = 5 [add 2]

++++ Quiz No.2 ++++++++++
2x+1=5x-8 [divide 2]

++++ Quiz No.3 ++++++++++
3x=6 [divide 3]
x = 2 [done]

++++ Quiz No.4 ++++++++++
2+3x=-2x+12 [divide 2]

++++ Quiz No.5 ++++++++++
2x-5=4-7x [divide 2]

++++ Quiz No.6 ++++++++++
6x-18=12 [divide 6]
x-3 = 2 [add 3]
x = 5 [done]

++++ Quiz No.7 ++++++++++
-2+4x=-16-3x [divide 2]

++++ Quiz No.8 ++++++++++
(x+2)/3=10 [done]

++++ Score: 2/8 (25%)
```

Fig. 2 Summary of quiz. The red steps are incorrect steps whereas the green steps are correct.

5 Authoring Learning by Teaching Environment

Applying SimStudent teachable agent and the LBT environment to other domains is easy and quick. Basically, to build a new LBT environment, one needs to create a Tutoring Interface and write the necessary background knowledge for SimStudent. The entire authoring process is rapid and easy, because SimStudent is originally developed as an intelligent plug-in component for Cognitive Tutor Authoring Tools (CTAT) [3] to help novice authors build their own Cognitive Tutors without heavy programming [2].

The Tutoring Interface used in the LBT environment is automatically taken from a Cognitive Tutor authored by CTAT. Depending on the subject domain to which the LBT environment is applied, additional background knowledge may need to be written with Java. There are some domain dependent components, such as examples and quiz items. These components are specified in a plane test file with fairly intuitive syntax. Interested readers can refer to the SimStudent project website (www.SimStudent.org) to learn more about how to apply SimStudent teachable agent to a new domain.

6 Discussion and Concluding Remarks

SimStudent is an inductive learner who genuinely acquires problem-solving skills from examples. The skills are represented as production rules, which is not directly sharable with the student who tutors SimStudent. The student must gauge SimStudent's proficiency in solving problems by observing SimStudent's behavior during problem solving. The student needs to diagnose SimStudent's errors and determine what problem should be posed next to remedy particular errors and/or reveal more errors. We anticipate that such a meta-level monitoring skills would enhance tutor learning. A preliminary lab study showed the effectiveness of learning equation solving by teaching SimStudent [10].

Our LBT learning environment provides researchers with an infrastructure to conduct a various controlled studies to explore the effect of tutor learning. We can, for example, test if the self-explanation effect applies to the tutor learning. We modify SimStudent so that it prompts the student to justify his/her tutoring activities. We can also test if an intervention of *meta-tutor* would facilitate the tutor learning. The impact of the initial proficiency level (and/or the learning capability) of SimStudent on tutor learning is another important factor that should be studied.

Using the proposed LBT learning, the researchers can various controlled studies to explore the effect of tutor learning easily and rapidly. Learning by teaching is a promising style of learning hence should be studied rigorously to establish robust cognitive theories of the tutor-learning effect.

7 Reference:

1. Roscoe, R.D. and M.T.H. Chi, *Understanding tutor learning: Knowledge-building and knowledge-telling in peer tutors' explanations and questions*. Review of Educational Research, 2007. **77**(4): p. 534-574.
2. Matsuda, N., W.W. Cohen, and K.R. Koedinger, *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)*. 2005, AAAI association: Menlo Park, CA. p. 1-8.
3. Aleven, V., et al., *The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains*, in *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, M. Ikeda, K.D. Ashley, and T.W. Chan, Editors. 2006, Springer Verlag: Berlin. p. 61-70.
4. Michie, D., A. Paterson, and J.E. Hayes, *Learning by teaching*, in *Proc. of Second Scandinabian Conference on Artificial Intelligence*. 1989: Tampere, Finland. p. 413-436.
5. Chan, T.-W. and C.-Y. Chou, *Simulating a learning companion in reciprocal tutoring systems*, in *Proceedings of the first international conference on Computer support for collaborative learning*. 1995. p. 49-56.
6. Leelawong, K. and G. Biswas, *Designing Learning by Teaching Agents: The Betty's Brain System*. International Journal of Artificial Intelligence in Education, 2008. **18**(3).
7. Nichols, D., *Issues in designing learning by teaching systems*, in *Proceedings of East-West Conference on Computer Technologies in Education*, P. Brusilovsky, et al., Editors. 1994.
8. Obayashi, F., H. Shimoda, and H. Yoshikawa, *Construction and evaluation of CAI system based on learning by teaching to virtual student*, in *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics*. 2000. p. 94-99.
9. Matsuda, N., et al., *A Computational Model of How Learner Errors Arise from Weak Prior Knowledge*, in *Proceedings of the Annual Conference of the Cognitive Science Society*, N. Taatgen and H. van Rijn, Editors. 2009, Cognitive Science Society: Austin, TX. p. 1288-1293.
10. Matsuda, N., et al., *Learning by Teaching SimStudent: Technical Accomplishments and an Initial Use with Students*, in *Proceedings of the International Conference on Intelligent Tutoring Systems*, J. Kay and V. Aleven, Editors. 2010 to appear, Springer: Heidelberg, Berlin.