An Individualized Web-based Algebra Tutor Based on Dynamic Deep Model Tracing

Dimitrios Sklavakis[1] and Ioannis Refanidis[1]

[1]University of Macedonia, Department of Applied Informatics,
Egnatia 156, P.O. Box 1591, 540 06 Thessaloniki, Greece
{dsklavakis, yrefanid}@uom.gr

Abstract. This paper describes the motivations and goals of the MATHESIS project which concerns the development of an intelligent authoring environment for cognitive math tutors. It also describes the first implemented component of the project, the MATHESIS algebra tutor, a cognitive web-based tutor for algebraic expressions' expanding and factoring. The tutor uses cognitive model tracing by dynamically generating the plausible steps, checking them against student's solution steps and intervening when errors occur. Additionally, the tutor monitors the student's mastery of knowledge from problem to problem, i.e. the various cognitive skills. The tutor will be used as a prototype for the development of an ontology that will contain all of the tutor's knowledge. This ontology will eventually guide the creation of the authoring tools that will make faster and easier the creation of other cognitive tutors.

Keywords: intelligent tutoring systems, model tracing, cognitive tutors, web based, authoring systems, ontologies.

1 Introduction

One-to-one tutoring has proven to be the most effective way of teaching. Professor B. S. Bloom and his colleagues [1] found that the average student under tutoring was about two standard deviations above the average of the conventional class (30 students to one teacher). The successful implementation of the one-to-one tutoring model by Intelligent Tutoring Systems (ITS) today poses the problem of how to develop ITS that provide the same tutoring quality with a human tutor. Cognitive Tutors, a special kind of model-tracing tutors developed at Carnegie Mellon University based on the ACT-R [2] theory of cognition and learning, have shown significant success in domains like mathematics and computer programming. However, Cognitive Tutors are hard to author. The development of the problem solving as well as the teaching knowledge requires considerable amounts of time and the recruitment of Ph.D. level scientists in education, cognitive science and artificial intelligence programming.

This paper presents the MATHESIS project which aims at developing authoring tools for Cognitive Tutors in mathematics as well as an initial product of the project, the MATHESIS Algebra Tutor, a cognitive tutor for mathematics in the domain of expanding and factoring basic algebraic expressions.

The rest of the paper is structured as follows: Section 2 describes the motivation and goals of the MATHESIS project. Section 3 presents the MATHESIS algebra tutor. Finally, Section 4 presents related work whereas Section 5 concludes the paper and poses future directions of research.

2 The MATHESIS Project: Motivation and Goals

The motivation for the MATHESIS project is the principled design and successful implementation of Cognitive Tutors in U.S. secondary education schools [3]. In order to better understand the MATHESIS project goals and components, a brief description of the ACT-R theory and the design principles it entails is given below.

2.1 The MATHESIS Project Motivation: Cognitive Tutors

Central to the ACT-R theory is the concept of *cognitive skill* defined as a set of *production rules* that describe the problem-solving steps. These production rules are IF-THEN rules which match the problem's goal(s) and current state and produce new sub-goals. For example, a production rule for monomial multiplication could be: IF the goal is to *multiply-monomials* THEN *multiply-coefficients* AND *multiply-mainParts*. Production rules form the *procedural knowledge* of a Cognitive Tutor. They operate (match) on *facts* which describe the problem's states (initial, intermediate, goal). Facts are implemented as lists of property-value pairs and form the *declarative knowledge* of a Cognitive Tutor. For example a fact could be: (current-operation multiply-monomials).

The declarative and procedural knowledge form the *cognitive model* of a Cognitive Tutor which implements the problem-solving knowledge of the domain to be taught. The *tutoring model* of a Cognitive Tutor is based on *model tracing* and *knowledge tracing*.

The model tracing algorithm matches the student's problem-solving steps with the ones produced by the cognitive model. As far as the student's solution remains on a correct path the tutor remains silent. Otherwise it provides feedback as soon as an error occurs. The tutor can also provide help for the correct step(s) upon student request. Therefore model tracing keeps track of the cognitive skills' acquisition inside a problem.

The knowledge tracing algorithm keeps track of cognitive skills' acquisition from problem to problem. The model tracing algorithm provides a percentage of skill acquisition and the knowledge tracing algorithm adjusts the proposed problems according to that percentage. In this way, Cognitive Tutors allow for self-pacing of the student through the curriculum.

2.2 The MATHESIS Project Goals

Despite Cognitive Tutors' efficiency, it is currently estimated that 1 hour of tutoring takes 200-300 hours of development [4]. The main reason for this is the *knowledge*

acquisition bottleneck: extracting the knowledge from the domain experts and encoding it into a program. Knowledge reuse appears as a necessity to overcome the knowledge acquisition bottleneck. Since expert knowledge and especially tutoring knowledge is so hard to create, re-using it is of paramount importance.

One widely used and quite promising technology for knowledge reuse is *ontological engineering* [5]. In the case of cognitive tutors, ontology engineering is the task of defining the cognitive model (facts, production rules) and tutoring model (user interface, model tracing and knowledge tracing) of the tutor and encode them in an ontology using specially designed environments for ontology management. This is the first research goal of the MATHESIS project. We believe that an efficient representation of a cognitive tutor's models in an ontology will provide a search space for the problem of cognitive tutor's authoring.

The second research goal is to develop the authoring tools that will help human authors search through this ontology space and therefore make their authoring faster and easier.

For the development and implementation of our research goals a bottom-up approach seems more appropriate. First, we need to implement a working prototype of a cognitive tutor. Then, the knowledge embedded in this tutor will be used to develop an ontology. Finally, based on the ontology we will develop the authoring tools whose purpose will be to guide the search through the ontology and help human authors.

3 The MATHESIS Algebra Tutor

The MATHESIS Algebra Tutor is a mathematics cognitive tutor for algebraic expressions' expanding and factoring. The domain of mathematics was chosen because it lends itself to bottom-up acquisition of cognitive skills and demands heavy reuse of them as well. In addition, adequate teaching expertise for developing the cognitive model of the tutor is available on behalf of one of the authors.

Three were the main issues that defined the overall architecture: a) the tutor interface should be web-based; we believe that the future of learning belongs to the world wide web and the tutor must be there, b) the model-tracing algorithm requires constant interaction between the cognitive model with the interface; therefore they should lie at the same side, that is the client side and c) the tutor should be able to be broken into pieces to produce the ontology and be reassembled back by the authoring tools.

The achievement of these requirements led us to implement the tutor using HTML for the interface and JavaScript for the cognitive and tutoring models. The primary interface element is Design Science's WebEq [6] Input Control applet, an editor for displaying and editing mathematical expressions. It provides the same functionality as Equation Editor for Microsoft Word. There are three such input controls, the algebraic expression, answering space and rough space input controls (Figure 1).

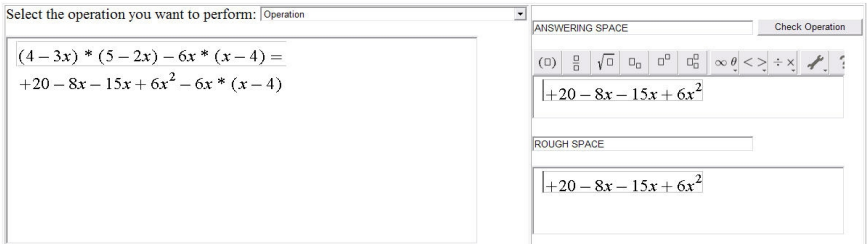

Fig. 1. The Algebraic Expression, Answering and Rough Space Input Controls

3.1 The Tutor's Cognitive Model

The top-level cognitive skills that the tutor teaches are the following: monomial multiplication, division and power, monomial-polynomial and polynomial-polynomial multiplication, parentheses elimination, collect like terms, identities (square of sum and difference, product of sum-difference, cube of sum and difference), factoring (common factor, term grouping, identities, trinomial).

These cognitive skills are further decomposed in more simple ones. As an example we will consider the *multiply-monomials* skill. This is decomposed in two others, *multiply-coefficients* and *multiply-mainParts*. The *multiply-mainParts* is further decomposed in finding common variables and adding their exponents and finding non common variables and copying their exponents. This decomposition is implemented through JavaScript functions that correspond to the production rules and JavaScript data structures (simple variables, arrays, custom objects) that correspond to the facts. There are also relevant functions for common error checking like omitting variables, or not adding the exponents of common variables.

3.2 The Tutoring Model: Deep Cognitive Model Tracing

Equipped with such a detailed cognitive model, the MATHESIS tutor is able to exhibit expert human-like performance. The tutor makes all the cognitive tasks explicit to the student through the structure of the interface. First, the tutor parses the algebraic expression and creates all the relevant facts (kind of operations and priorities of them). The student must select a part of the algebraic expression and then select the operation he/she thinks corresponds to that part. Then the tutor, based on the parsed knowledge described above, checks the proposed operation against the selected expression. For example, the tutor checks if the student has selected only one operation or more, if the operation selected has the right priority to be performed, if the proposed operation matches with the part of the expression selected by the student. Only then the tutor proceeds to perform the operation.

In this stage, the tutor guides step by step the student with appropriate messages. Of course, in every step the tutor calculates the result, gets the student's answer and checks it for correctness. If any partial result is incorrect then the tutor displays the appropriate messages and asks again for that result in order to proceed. In the poly-

nomial multiplication $(4-3x)*(5-2x)$, the tutor prompts the student for each one of the four monomial multiplications that must be performed, i.e. $4*5$, $4*(-2x)$, $-3x*5$ and $-3x*(-2x)$. For each one of the monomial multiplications the tutor behaves as if it was teaching the monomial multiplications as separate exercises, performing all the necessary cognitive tasks and checks. That's what we call *deep cognitive model tracing*. It must be pointed out that this deep cognitive model tracing is what makes one-to-one tutoring so effective and it is not an easily implemented feature even for a cognitive tutor. It is possible only with detailed cognitive task analysis which has knowledge reuse as a primary design parameter.

3.3 The Student Model: Knowledge Tracing

Based on such a detailed cognitive model and the deep model tracing feature, the MATHESIS algebra tutor keeps a detailed *student model*, that is which skills the student has mastered and to what extent from problem to problem. For each supported cognitive skill, e.g. monomial multiplication, the tutor keeps counters for the correct and incorrect answers of the student. With this simple mechanism, the tutor keeps track of the mastery level of each cognitive skill as a percentage calculated by the formula

$$mastery\ level = \frac{correct\ answers}{correct + incorrect\ answers} * 100\% \quad (1)$$

This percentage is time-stamped, i.e. the tutor keeps the date when a percentage changes, creating an accurate image of the mastery level change over time for every cognitive skill. It is important to stress out that the tutor updates the student model for all the cognitive skills that are present in a specific exercise. For example, when the student has to perform a *multiply-monomials* task, he/she must perform many *multiply-monomials* tasks. The tutor will update and time-stamp the mastery levels for this skill too. This behaviour is what we call *broad knowledge monitoring* and is a direct consequence of the *deep cognitive model tracing* feature of the tutor.

Although such a detailed, broad and dynamic student model gives the ability to the tutor to be highly adaptive as to what must be taught to every individual student, for the moment, the student model is just presented to the student and to the human tutor(s) that are responsible for assessing the students' knowledge. It is in our plans to design and implement a module that would use the student model to automate the selection of exercises to present to the student according to his/her mastery level of the various skills and the skills covered by each exercise.

4 Related Work

Despite their performance, Cognitive Tutors are proprietary, stand-alone applications that provide tutoring for a pre-programmed set of problems and of course they have the same high-cost demand in time and human resources [7].

To overcome these limitations, Carnegie Mellon University researchers have been developing the Cognitive Tutor Authoring Tools (CTAT) [8], a set of software tools that intend to make cognitive tutors' development easier and faster. The tools mainly support the authoring of *example-tracing tutors*. In these tutors, instead of writing production rules the author records the correct (or incorrect) answer for every step in the solution and the tools match these answers with the student's answers. Based on these tutors the tools provide debugging of the production rules that the author has to find and write by himself. In addition, the rules are stored in files from where they must manually be loaded and executed. Therefore the authoring knowledge remains isolated and en-coded. It is the ultimate goal of the MATHESIS project to re-code and open that knowledge through an ontology.

5 Discussion and Further Work

The MATHESIS algebra tutor has not been evaluated in a real school environment since it is still under development and testing. However, we got positive feedback when it was demonstrated to a few teachers of mathematics in Greek secondary education. Of course, being a research prototype, it needs more development and testing. Significant tutoring issues, like the granulation of the tutoring steps and pedagogical issues, like how to use the student model, are open.

What is important is the fact that the tutor's overall architecture and design will allow us to proceed to the second step of the MATHESIS project and develop an ontology that will contain all the knowledge now embedded to the HTML and JavaScript code. The ontology will make this authoring knowledge open and therefore reusable.

References

1. Bloom, B.S.: The 2 Sigma Problem: The Search of Methods for Group Instruction as Effective as One-to-One Tutoring, Educational Researcher, Vol. 13, No. 6, 4--16 (1984)
2. Anderson, J.R., Corbett, A.T., Koedinger, K.R., Pelletier, R.: Cognitive Tutors: Lessons Learned, The Journal of the Learning Sciences, 4(2), 167—207 (1995)
3. Carnegie Learning, http://www.carnegielearning.com/products.cfm
4. Murray, T.: An overview of intelligent tutoring system authoring tools: Updated Analysis of the State of the Art, In: Murray, Ainsworth, and Blessing (eds.), Authoring Tools for Advanced Technology Learning Environments, pp 491--544, Kluwer Academic Publishers, Netherlands (2003)
5. Aitken, S.J., Sklavakis, D., Integrating problem solving methods into CYC, In: Dean, T. (ed.), Proceedings of the International Joint Conference on Artificial Intelligence 1999, pp 627--633, Morgan Kaufman Publishers, San Francisco (1999)
6. Design Science, http://www.dessci.com/en/products/webeq/
7. Aleven, V., McLaren, B., Sewall, J., Koedinger, K.R.: The Cognitive Tutor Authoring Tools (CTAT): Preliminary Evaluation of Efficiency Gains, In: Intelligent Tutoring Systems 2006, LNCS, vol. 4053, pp 61--70, Spinger, Berlin, (2006)
8. Cognitive Tutors Authoring Tools (CTAT), http://ctat.pact.cs.cmu.edu/