International Journal of Artificial Intelligence in Education
https://doi.org/10.1007/s40593-023-00328-3

ARTICLE

A Step-Based Tutoring System to Teach Underachieving Students How to Construct Algebraic Models

Kurt VanLehn[1] · **Fabio Milner**[1] · **Chandrani Banerjee**[1] · **Jon Wetzel**[1]

Accepted: 12 January 2023

Abstract

An algebraic model uses a set of algebraic equations to describe a situation. Constructing such models is a fundamental skill, but many students still lack the skill, even after taking several algebra courses in high school and college. For such students, we developed instruction that taught students to decompose the to-be-modelled situation into schema applications, where a schema represents a simple relationship such as distance-rate-time or part-whole. However, when a model consists of multiple schema applications, it needs some connection among them, usually represented by letting the same variable appear in the slots of two or more schemas. Students in our studies seemed to have more trouble identifying connections among schema applications than identifying the schema applications themselves. We developed several tutoring systems and evaluated them in university classes. One of them, a step-based tutoring system called OMRaaT (One Mathematical Relationship at a Time), was both reliably superior ($p = 0.02$, $d = 0.67$) to baseline and markedly superior ($p < 0.001$, $d = 0.84$) to an answer-based tutoring system using only commercially available software (MATLAB Grader).

Keywords Intelligent tutoring system · Algebraic model construction · Algebra story problem solving · Algebra word problem solving

✉ Kurt VanLehn
kurt.vanlehn@asu.edu

Fabio Milner
milner@asu.edu

Chandrani Banerjee
cbanerj1@asu.edu

Jon Wetzel
jon.wetzel@asu.edu

[1] Arizona State University, Tempe, AZ, USA

Published online: 31 January 2023

The Research Problem and Prior Work on it

The Research Problem: Teaching Students to Construct Algebraic Models

Constructing models is a fundamental and important skill. According to the Next Generation Science Standards (NGSS, 2013), "developing and using models" is one of 8 key scientific practices. According the Common Core State Standards for Mathematics (CCSSM) (Common Core State Standards for Mathematics, 2011), "modeling with mathematics" is one of its 8 key mathematical practices.

Students are introduced to model construction with arithmetic story problems in primary school, and then algebraic story problems in secondary school. Both are notoriously difficult. However, not all students who take these courses learn how to construct models. For instance, across several years of teaching a junior-level university engineering course on model construction, the first author found that about half the entering class could not construct algebraic models of stories like the one shown in Fig. 1. Thus, the research problem addressed here is teaching algebraic model construction to university students who somehow failed to acquire this important skill in earlier classes.

A Theory of Algebraic Model Construction and Instruction Based on it

Several researchers have applied Kintch's theory of text comprehension to model construction (Cummins et al., 1988; Fuchs et al., 2010, 2019; Jitendra et al., 2015; Kintsch & Greeno, 1985; Marshall, 1995; Nathan et al., 1992; Riley & Greeno, 1988; Xin et al., 2005). The theory posits that students construct equations by matching schemas against their understanding of the story. Each match of a schema fills slots of the schema and produces an equation. The part of the story that matches the schema is called a mathematical relationship. Nathan et al., (1992) observed that some relationships were obvious to students and some were not. Schemas, which are a central construct of many cognitive theories of

> Shortly after an F-35 fighter jet passes over some militants, they fire an FIM-92 Stinger missile at the plane. The plane flies at full speed, 537 m/s. The missile flies at its full speed, 750 m/s. How long (in seconds) must the missile travel in order to catch up with the plane? Assume the militants fire the missile 6 seconds after the jet passes over them. Ignore the fact that a Stinger missile runs out of fuel after 10 seconds.

Fig. 1 A story problem

understanding, have a set of conditions that both recognize when the schema applies and fill its slots. Schemas also have a set of inferences to be made when a schema applies. For example, the *Overtake* schema can be rendered informally as:

- *Slots*: Object1, Object2, Path
- *Conditions*: Object1 and Object2 both travel along Path. They start at the same point. Object1 starts after Object2. Object1 eventually overtakes Object2.
- *Inferences*: Object1's average speed > Object2's average speed. At the moment when Object1 overtakes Object2, they are at the same point along Path. The distance travelled by Object1 = the distance travelled by Object2. The duration of Object1's trip < the duration of Object2's trip.

Experts often have large schemas that match whole stories. For example, an expert might recognize the problem of Fig. 1 as a delayed start overtake problem, and write.

- $537 * Tplane = 740 * (Tplane + 6)$; distances are equal, but durations are offset

Although such composite schemas are handy, several studies showed that atomic (i.e., not composite) schemas like Comparison and Motion (i.e., distance-rate-time) were both less error-prone and easier to learn (Blessing & Ross, 1996; Gerjets et al., 2004, 2006). Thus, researchers have focused on teaching only atomic schemas.

The equations below illustrate a model of the story in Fig. 1 using only atomic schemas.

- $Dplane = 537 * Tplane$; obvious application of the Motion schema
- $Dmissile = 740 * Tmissile$; obvious application of the Motion schema
- $Dplane = Dmissile$; nonobvious application of the Overtake schema
- $Tplane = Tmissile + 6$; nonobvious application of the Comparison schema

Many methods for teaching model construction have been tried. Table 1 lists some major ones detailed in VanLehn (2013). Perhaps the most well-developed and successful method is Schema-Based Instruction (SBI) (Fuchs et al., 2003, 2004a, b, 2009, 2010, 2019; Hutchinson, 1993; Jitendra et al., 2007, 2009, 2011, 2013, 2015; Xin et al., 2001, 2005). It is founded on the first two methods listed in Table 1. It teaches arithmetic (not algebraic) model construction. Moreover, most SBI studies used problems that contained only one mathematical relationship, whereas most algebraic story problems have several mathematical relationships. Thus, adapting SBI to algebra requires developing instruction to teach students (a) how to decompose a story into multiple atomic relationships, (b) how to recognize nonobvious relationships and (c) how to link relationships together by recognizing when two relationships referred to the same quantity and thus their equations should share a variable.

Table 1 Methods for teaching model construction

Teaching atomic schemas explicitly (Fuchs et al., 2003, 2004a, b, 2009, 2010, 2019; Hutchinson, 1993; Jitendra et al., 2007, 2009, 2011, 2013, 2015; Xin et al., 2001, 2005)
Network representation of equations (Bridewell et al., 2006; Chang et al., 2006; Derry & Hawkes, 1993; Fuchs et al., 2003, 2004a, b, 2009, 2010, 2019; Hutchinson, 1993; Jitendra et al., 2007, 2009, 2011, 2013, 2015; Joolingen et al., 2005; Löhner et al., 2003, 2005; Marshall, 1995; McArthur et al., 1989; Metcalf et al., 2000; Pauli & Reusser, 1997; Reusser et al., 1993; Willis & Fuson, 1988; Xin et al., 2001, 2005)
Decomposition of a system into subsystems (Ramachandran, 2003; Heffernan et al., 2008; Heffernan & Koedinger, 1997)
Displaying the meaning of variables (Avouris et al., 2003; Forbus et al., 2005; Metcalf et al., 2000)
Feedback/hints on the student's steps during problem solving (Arnau et al., 2013; Chi & VanLehn, 2008, 2010; Leelawong & Biswas, 2008; Zhang et al., 2014)
Adaptive task selection (Arroyo, 2000; Beal et al., 2010; Beck et al., 2000; Koedinger et al., 2008)
Concrete articulation strategy (Heffernan, 2003; Heffernan & Croteau, 2004; Heffernan & Koedinger, 1997; Heffernan et al., 2008; Koedinger & Anderson, 1998; McArthur et al., 1989)
Using bars to represent quantities and their relationships (Looi & Tan, 1996, 1998; Munez et al., 2013)
Using animations as the predictions of the model (Gould & Finzer, 1982; Nathan et al., 1992)
Student or system restates problem in mathematical English (Heffernan & Koedinger, 1997; Heffernan et al., 2008)
Self-explanation of worked examples (Cooper & Sweller, 1987; Corbett et al., 2003, 2006; Heffernan et al., 2008; Renkl et al., 1998)
Training students to recognize irrelevant quantities (Cook, 2006; Fuchs et al., 2010; Heffernan & Koedinger, 1997)
Systematically varying the surface features of the systems (Fuchs et al., 2003, 2004a, b, 2009, 2010; Renkl et al., 1998)
Asking students to find values for all unknowns instead of just one (Sweller et al., 1983)
Feedback/hints on the model (Biswas et al., 2005; Bravo et al., 2009; Zhang et al., 2014)
Reflective debriefings (Connelly & Katz, 2009; Katz et al., 2003, 2007)
Answering student questions (Anthony et al., 2004; Beek et al., 2011; Corbett et al., 2005; Leelawong & Biswas, 2008; Segedy et al., 2012)
Students explaining their model (Heffernan et al., 2008; Metcalf, 1999; Metcalf et al., 2000)
Qualitative-first model construction (Kurtz dos Santos & Ogborn, 1994; Bredeweg et al., 2010; Mulder et al., 2010, 2011)
Model progressions (Jong et al., 1999; Quinn & Alessi, 1994; Swaak et al., 1998; White, 1984, 1993; White & Frederiksen, 1990)
Teachable agents and reciprocal teaching (Biswas et al., 2005; Chan & Chou, 1997; Chase et al., 2009; Pareto et al., 2011; Reif & Scott, 1999)
Mental execution of models (Kurtz dos Santos & Ogborn, 1994; Löhner, 2005)
Generic models (Bredeweg & Forbus, 2003; Bridewell et al., 2006; Marshall et al., 1989)
Gamification (Schwartz et al., 2009)

Our Prior Work on Tutoring Algebraic Model Construction

Our first three years of work are reported in an earlier publication (Vanlehn et al., 2020), so this summary will be brief.

Assessment Story problems, like the one in Fig. 1, end by asking for the value of a quantity. Some students have developed numerical methods for calculating such numbers without writing equations (Koedinger & Nathan, 2004; Koedinger et al., 2008). This may explain how they could pass algebra courses without learning how to construct models. Thus, to assess their skill at model construction, we developed a simple assessment system called the Solver (Fig. 2). Students type equations into a textbox, one per line. When they click a button, the Solver either posts an error message or solves the equations and reports values for all the variables. If the students think these values solve the problem, then they copy the equations and submit them as their solution to the problem. Unlike similar systems available as software or on calculators like the TI-89, the Solver only allows students to include a number in their equations if that number is mentioned in the story. This prevents them from using numerical strategies for solving story problems.

Population We began working with students who had failed College Algebra and were taking a remedial course at an open-admission university. Initially, the researchers (which included two of the course instructors, Banerjee and Milner) tutored students in small groups or individually for about 20 h per student. We called these "boot camps" for algebraic modeling. The boot camps started with paper and pencil then gradually started using the tutoring system we were developing. Attrition was high. Only the most committed students were willing to spend 20 or even 10 h attending our boot camp, even when the pay was high, and sessions could be done remotely. Thus, we switched to teaching students in the first author's class on

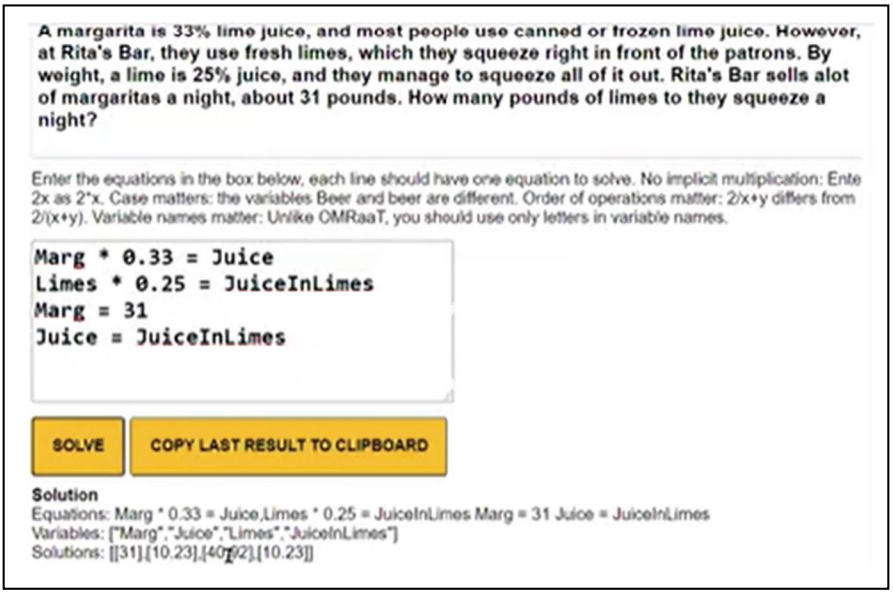

Fig. 2 A problem solved correctly in the Solver. The bold text in the box was entered by the student, who then pressed the Solve button

computational modelling, which is taken by 3rd- and 4th- year engineering majors. The boot camp instruction was converted to a 2-week module in the course. Just before the module began, we tested all students using the Solver. As with the remedial College Algebra students, some students answered very few problems correctly.

Evolution of the Design As mentioned earlier, SBI is a successful method for teaching construction of models consisting of a single arithmetic equation. SBI is based on the first two methods of Table 1. In order to extend it to models consisting of multiple algebraic equations, we added several methods with good empirical support. Our design was based on the first 5 methods in Table 1.

Over a series of 5 studies (formative evaluations), we experimented with variations of: the model notation, feedback policies, the schemas to be taught, and the instructional sequence. We often made changes in the middle of the instruction. Changes were often based on student suggestions. This iterative development method is sometimes called design-based research (Reimann et al., 2011).

We ended up with a node-link intermediate representation that was somewhat similar to the one used in SBI instruction. Figure 3 shows an example. The oval nodes represent quantities and the rectangular nodes represent atomic schema applications. We named the tutoring system TopoMath because it used the topology of the graph to represent algebraic models.

TopoMath was an editor with three commands: (1) To enter a schema application, students selected a schema name (e.g., Compare, Motion) then selected an entity that differentiated this schema application from others (e.g., plane vs. missile). This caused TopoMath to create a rectangular equation node and several gray oval variable nodes. (2) The variable nodes remained gray until the student edited them to fill in units and possibly values. Students could also cause two variable nodes to merge by editing the name of one to be the name of the other. (3) Students could drag nodes.

Students were taught that when they could not identify mathematical relationships, they should drag similar unknown variable nodes to be close to each other, as shown in Fig. 3. The proximity of the nodes was intended to cue recognition of non-obvious relationships. Thus, Fig. 3 should help them recognize that $DPlane = DMissile$ and $TPlane = TMissile + 6$.

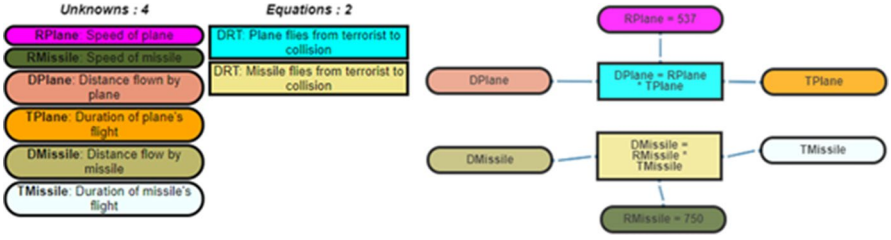

Fig. 3 A partial TopoMath model for the story in Fig. 1

Throughout this problem-solving process, the tutoring system gave immediate, minimal feedback (green for correct; red for incorrect). In order to discourage guessing, on a third incorrect response, the system made the correct entry and colored it yellow. The yellow persisted when a node was closed and thus could be easily seen by the instructor or grader.

Summative Evaluation To evaluate the TopoMath instruction, we used a regression discontinuity design. Students first took a pretest using the Solver. They were then split into a treatment group and a no-treatment group based on whether their pretest score was below a cutoff score (treatment group) or above it (no-treatment group). The treatment group was required to use the TopoMath instruction for about 2 weeks. To be fair to the non-treatment students, TopoMath was available to them too, albeit not required. Both groups then took a posttest on the Solver. The posttest was isomorphic to the pretest; only the cover stories were changed. Students' scores on the posttest were part of their grade for the course.

Figure 4 shows the results as a regression of posttest scores against pretest scores. The pretest cutoff score was 0.61, and as expected, we see a discontinuity at that

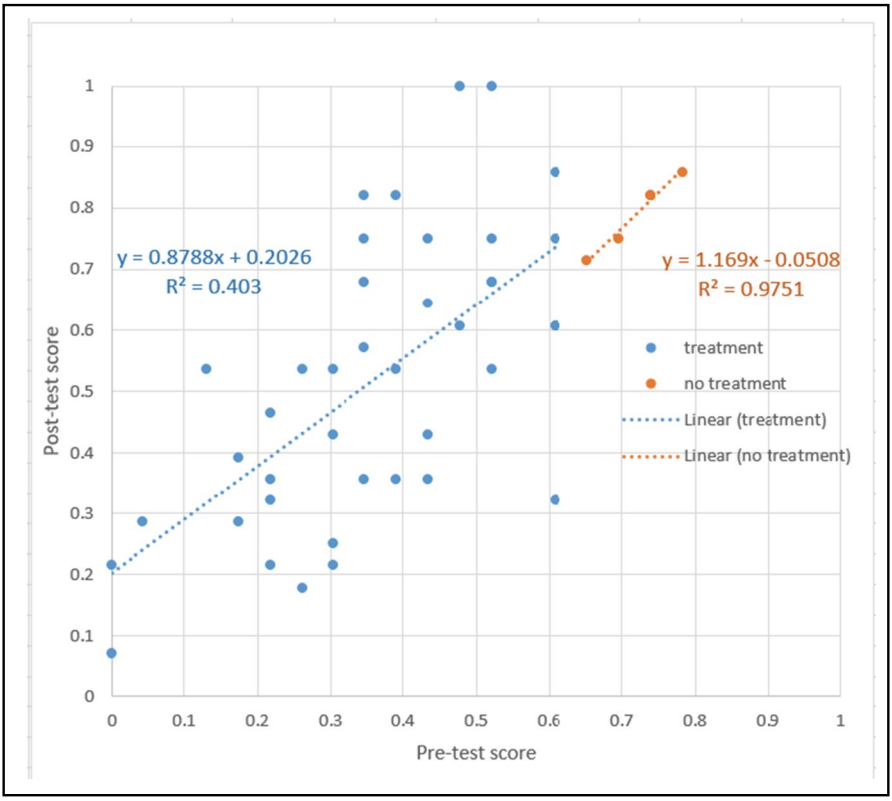

Fig. 4 Results of the evaluation of TopoMath

point. However, this two-line model was not significantly different from a single-line model ($p = 0.74$). Thus, although the treatment may have a benefit, it was too small relative to the large scatter in scores.

Unfortunately, our decision to make TopoMath optional for the no-treatment group motivated most of them to use TopoMath. Only 5 students never used it, so only they are included in the analyses of Fig. 4. (Two students had identical pretest and posttest scores, so they are shown as one point.) Thus, our decision to be fair to the no-treatment group dramatically reduced the power of the study.

OMRaaT and the OMRaaT module

Intent on increasing the benefits of our instruction, we again tutored students using first pencil and paper and then a mock-up in Excel of a tutoring system. These sessions led us to design a new tutoring system and instruction from scratch. They differed from TopoMath and its instruction in two major ways.

(1) The network representation could only show a small amount of text inside each node, so many of the details used for establishing the meaning of the node could only be seen by clicking on it. We changed the notation so that no information is hidden.
(2) The TopoMath instruction taught students that variables denoted quantities, which is the classic semantics of variables. For example, the variable "Dplane" denotes the distance the plane flew and "DMissile" denotes the distance the missile flew. This made merging nodes confusing, because students didn't know which node to edit. For example, in Fig. 3, the result of merging the two variable nodes on the left was labelled either DPlane or DMissile, depending on which node was edited. The instruction was changed to remove the concept of a variable denoting a quantity and instead treat variables as names for slots. To replace merging of nodes, a new schema was added: Equality. It indicated that two slots have the same value. The Equality schema included a description slot for justifying why the two slots have equal values. For example, when equating the distances travelled by the plane and the missile, the justification was "travelled approximately the same path."

In terms of the methods listed in Table 1, the new tutoring system retained 4 of the 5 methods and eliminated one: Network representation of equations. It added a new method: Explicitly represent slot sharing. The new tutoring system was named OMRaaT: an acronym for One Mathematical Relationship at a Time.

Figure 5 shows a solved problem in OMRaaT. Each row is a schema application. The boxes are slots. When students select the name of a schema, a new row is added to the table with empty slots labelled by the text above them. The first few slots describe the schema application. The Motion schema (first two

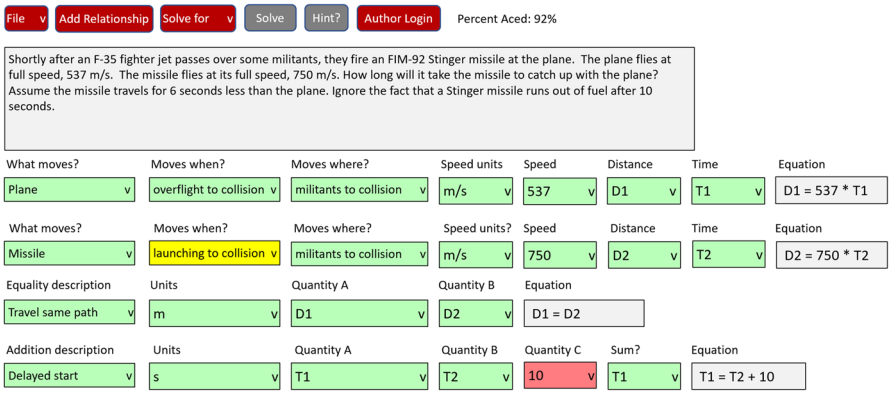

Fig. 5 A solved problem in OMRaaT

rows of Fig. 5) has 4 description slots; the Equality schema (third row) has 2; the Addition schema (last row) has 2. Students fill a description slot by selecting from a menu including all possible description slot fillers.

When the student finishes filling the description slots of a schema application, the slots turn red (incorrect) or green (correct). The student then tries to correct the red slots. On the third incorrect attempt, the slot turns yellow and shows the correct entry. The yellow coloring and the delayed feedback are intended to discourage guessing. Also, the percentage of slots filled correctly on the first attempt is displayed at the top of the window (e.g., "Percent Aced: 92%").

After all the description slots of a schema application have been filled correctly, the student can fill the remaining slots, which are called *quantity* slots. For the Motion schema, there are 3 quantity slots, for distance, rate and time. To fill a quantity slot, students select from a menu that has numbers mentioned in the story (e.g., 537, 750, 6 and 10) and "Unknown." If they select Unknown, OMRaaT invents a variable name that is unique to the slot. Thus, the variables denote slots. When all the quantity slots have been filled, the student gets red/green feedback.

We reduced the set of problems slightly so that only four schemas were needed: Motion, Mixture, Addition and Equality. The Mixture schema represents relationships such as "rum is 40% alcohol" and has slots for a mixture, an ingredient and a proportion. We hypothesize that it would be easy for students to learn more schemas after they acquire the skill of decomposing the problem into atomic relationships and determining when schemas share slots.

We originally intended that all connections among schemas would be represented with the Equality schema. However, this entailed entering too many Equality schemas. For the problem of Fig. 5, three more Equality schema applications would be needed. Thus, to simplify the models and reduce problem solving time, we allowed the Addition schema to refer to existing slots, as shown in Fig. 5. To make this work smoothly, we implemented a suggestion of Nathan (Nathan et al., 1992) and

required student to first enter obvious schemas (Motion and Mixture) before entering any nonobvious schemas (Equality and Addition).

When students finish correctly entering all the schema applications required, they click the "Solve for:" button to select one of the variables, and then click the Solve button, which pops up a window showing the numerical value of the solved-for variable.

The overall instruction was implemented as a module in the university's LMS, Canvas. The OMRaaT module comprised 15 passive pages, which had only videos or text, and 47 active pages, each with a problem to solve. Problems were solved on either OMRaaT or the Solver. After finishing a problem on OMRaaT or the Solver, students pasted their model into Canvas. Neither Canvas nor the instructor graded the submissions. The course instructor was available via email and zoom; there was no course TA nor tutoring available. The students had 19 days to complete the OMRaaT module. On the 20^{th} day, there was a module exam, which also served as the post-test for this experiment.

The OMRaaT module was organized as a sequence of 5 sections, each teaching a new schema or concept. Each section began with an explanation and a worked example; each ended with a Solver problem; in between, the students did OMRaaT problems of increasing complexity.

Solver problems were included because we wanted to make sure that students transferred the OMRaaT reasoning to the Solver, which provided no feedback on correctness and little strategic guidance. The module's Solver assignments emphasized three key components of the OMRaaT strategy: (1) Only write atomic equations, that is, equations with just one algebraic operator: +, -, * or /. (2) Use meaningful variable names instead of x, y and z. (3) Only include numbers mentioned in the problem statement. The Solver enforced the third constraint, but it let students violate the other two if they wished. The Canvas Solver pages emphasized that they should always obey all three constraints.

The Canvas gradebook showed each problem-solving page as a separate column. This "dashboard" allowed the instructor to track student progress and nag students who were falling behind.

It is perhaps worth mentioning how we managed to get Canvas to act like a dashboard-laden ITS "for free." To avoid installation issues, OMRaaT was a web app. In principle, we could have used LTI to integrate it with Canvas. However, to avoid the university's requirement for security scans of software integrated with Canvas, OMRaaT communicated with Canvas via files. To solve a problem, students downloaded a file from the Canvas assignment page, uploaded it to OMRaaT, solved the problem, downloaded the modified file, and then uploaded it to the Canvas page. This hack saved us many months trying to pass a security scan.

An Evaluation of the OMRaaT Module

To evaluate the OMRaaT module, we again used a regression discontinuity design. This time, students above the cutoff on the pretest (the no-treatment group) were prevented from taking the OMRaaT modules. As in the TopoMath evaluation, students below the cutoff (the treatment group) were required to take the module.

To be fair to the students in both conditions, algebraic modeling was taught twice bracketed by tests. The sequence was: (1) pretest, (2) OMRaaT module for the treatment group only, (3) mid-test, (4) Solver-based instruction on algebraic modelling for all students, and (5) posttest. Students' scores on the mid-test and posttest counted towards their grade, whereas scores on the pretest were only used for placement in the treatment or no-treatment condition.

All 3 tests were given on the Solver. Before taking the pretest, students were introduced to it and solved several very simple problems using it.

The pretest and mid-test were isomorphic. Problems appeared in the same order. For every problem on the pretest, the corresponding problem on the mid-test had the same algebraic structure but different words. For example, one problem talked about discounts at Whole Foods and the other talked about discounts at Kohls. There were 4 problems with one schema application, 4 problems with two schema applications, 1 problem with three schema applications, 2 problems with four schema applications and 1 problem with five schema applications.

Unfortunately, scores on the pretest were at ceiling on 3 of the problems. That is, on 2 problems with a single schema and 1 problem with two schemas, 88% or more of the students got the problem correct on the pretest, which was well above the average on the other pretest problems (47%). Thus, those three problems were eliminated from the data analysis. This left both tests with 10 problems each.

Points were assigned to test problems according to the number of schema applications required for their solution. For example, a problem with 3 schema applications was worth 3 points. Thus, the maximum score on each test was 24 points. The maximum number of points in the overall course was 455, and a student's course grade was proportional to the number of points the student scored during the course.

The cutoff for dividing the students into treatment and no-treatment groups was the median of the pretest scores of all the students in the class. However, only 50 students took both tests and consented to have their data used. Of these, 28 scored below the cutoff on the pretest and thus were in the treatment group. The remaining 22 were in the no-treatment group.

The treatment group students could access the OMRaat module in Canvas. The no-treatment group could not access it. The no-treatment group had no algebraic modeling instruction or other activities during the 20-day treatment period. However, during this period, the whole class met as usual twice a week for lectures on an unrelated topic (human problem solving) accompanied by a few pages of reading.

The treatment group scored a half point for each problem solution they submitted to Canvas, regardless of whether it was correct or not. The no-treatment students were awarded the maximum of these points.

Although 27 students in the treatment group did almost all the problems (average number of completed problems: 46.0 out of 47), one student in the treatment group did only 4. This student was excluded from the treatment group during data analysis.

We first analyzed the mid-tests to see whether students obeyed the key OMRaaT strategy constraints, which were: (1) Only write atomic equations, that is, equations with just one algebraic operator. (2) Use meaningful variable names. Students were coded as obeying the strategy when greater than 50% of their solutions obeyed both constraints. We found that 4 treatment students did not use the strategy on the

mid-test, so they were excluded from further data analysis. Although they may have mastered the OMRaaT skills, they apparently chose not to use them on the mid-test, so we can't assess their OMRaaT learning gains. This left 23 students in the treatment group for further data analysis.

Figure 6 shows the regression plot for the two groups. The solid line is the diagonal, where mid-test scores equal pretest scores. All the treatment students gained: their mid-test scores were larger than their pretest scores, often by large amounts. Some (59%) of the no-treatment students also gained, which could be due to a test–retest effect or gaining familiarity with the Solver. However, for a significant proportion (36%) of the no-treatment students, the scores actually decreased from pretest to mid-test.

However, the success of a regression discontinuity design hinges on whether the regression lines of the two groups (shown in blue and red) fit the data better than a regression line for the union of the two groups (shown in green). One cannot go by the R^2 values (shown in Fig. 6) because they are derived from 3 different datasets. Thus, we subtracted the cutoff value (11.5) from the pretest scores, then fit this equation to all the data:

$$\text{MidTest} = \text{A} + \text{B} * \text{Pretest} + \text{C} * \text{Group} + \text{D} * \text{Group} * \text{Pretest}$$

where Group is 1 for the treatment group and 0 for the no-treatment group. The Group variable means that coefficients C and D affect only treatment data points, establishing a second line that is above the line established by A and B. The best fit found:

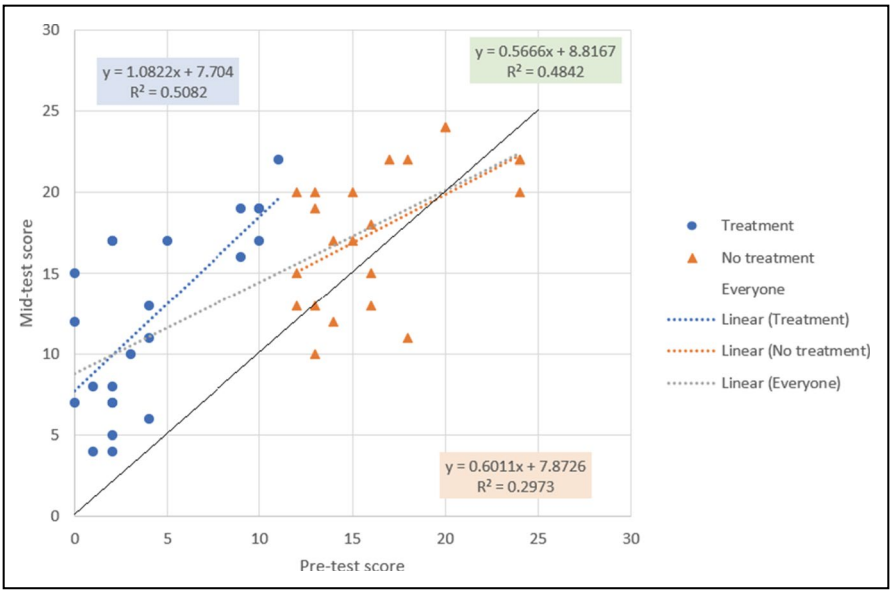

Fig. 6 Results of the OMRaaT module evaluation

- A = 14.8, $p < 0.001$
- B = 0.61, $p = 0.008$
- C = 5.36, $p = 0.023$
- D = 0.48, $p = 0.129$

Because C is reliably different from zero, the two-line model fits better than the one-line model. The coefficients compare the regression line for the treatment group alone to the regression line for both groups. Because D was almost zero, the lines' slopes were the same. Because C was reliably different from zero, the treatment seems to have raised scores by a constant amount. Thus, we conclude that the OMRaaT treatment worked in that it raised mid-test scores reliably above what they would otherwise have been.

In order to estimate how much the treatment improved mid-test scores, we used the main line of the model (MidTest = 14.8 + 0.601*Pretest) to predict mid-test scores if the students had not taken the OMRaaT treatment. Comparing the actual scores to the predicted scores yielded an effect size of d = 0.67. To interpret this effect size as recommended by Lipsey et al. (2012): 61% of the actual scores exceeded the mean of the predicted scores.

A Conventional Answer-Based Tutoring System

This section presents a comparison of OMRaaT to an answer-based tutoring system. There were two motivations for this study. One was to test the hypothesis that step-based tutoring systems tend to be more effective than answer-based tutoring systems (VanLehn, 2011). The second was that the Solver developed bugs which could not be fixed because funding for the project was finished, so we had to switch to other equation-solving software.

The new tutoring system used only commercially available software, namely MATLAB Grader. MATLAB Grader tests student programs using inputs and outputs. For each problem, the instructor supplies a set of tests. A test consists of inputs and outputs. The student's program must have inputs and outputs. The test's inputs are fed to the student's program, and the program's outputs are compared to the test's outputs. MATLAB Grader is similar to Gradescope and many other autograders. Unlike Gradescope and others, MATLAB can solve sets of equations.

In order to use MATLAB Grader, the modeling problems had to be modified slightly so that they specified inputs and outputs. For an example, consider the problem of Fig. 1:

> Shortly after an F-35 fighter jet passes over some militants, they fire an FIM-92 Stinger missile at the plane. The plane flies at full speed, 537 m/s. The missile flies at its full speed, 750 m/s. How long (in seconds) must the missile travel in order to catch up with the plane? Assume the militants fire the missile 6 seconds after the jet passes over them. Ignore the fact that a Stinger missile runs out of fuel after 10 seconds.

The new problem, with edits in italics, is:

> Shortly after an F-35 fighter jet passes over some militants, they fire an FIM-92 Stinger missile at the plane. The plane flies at full speed, 537 m/s. The missile flies at its full speed, 750 m/s. *Create a model that outputs the time (in seconds) that* the missile must travel in order to catch up with the plane. Assume the militants fire the missile *a few seconds* after the jet passes over them. *Your model should input the duration of this delay in seconds*. Ignore the fact that a Stinger missile runs out of fuel after 10 seconds.

Figure 7 shows a problem as presented to students in MATLAB Grader. The problem text is at the top. Next is a link to a video of the instructor solving the problem. A template for entering the solution appears in a box. Students are told to replace the symbols that are in all capitals. They cannot edit the lines that are gray. After the box comes some code for testing the function. When students click the Run Function, those tests are run, and the outputs are printed. If the function has a bug, a MATLAB error message appears instead. Not shown is a Submit button that tests the function on new inputs, and reports whether the function is correct or not.

Figure 8 shows a solution to the problem. Although MATLAB Grader gives full marks to any function whose input–output behavior passes the tests, students were told to make their equations obey three rules:

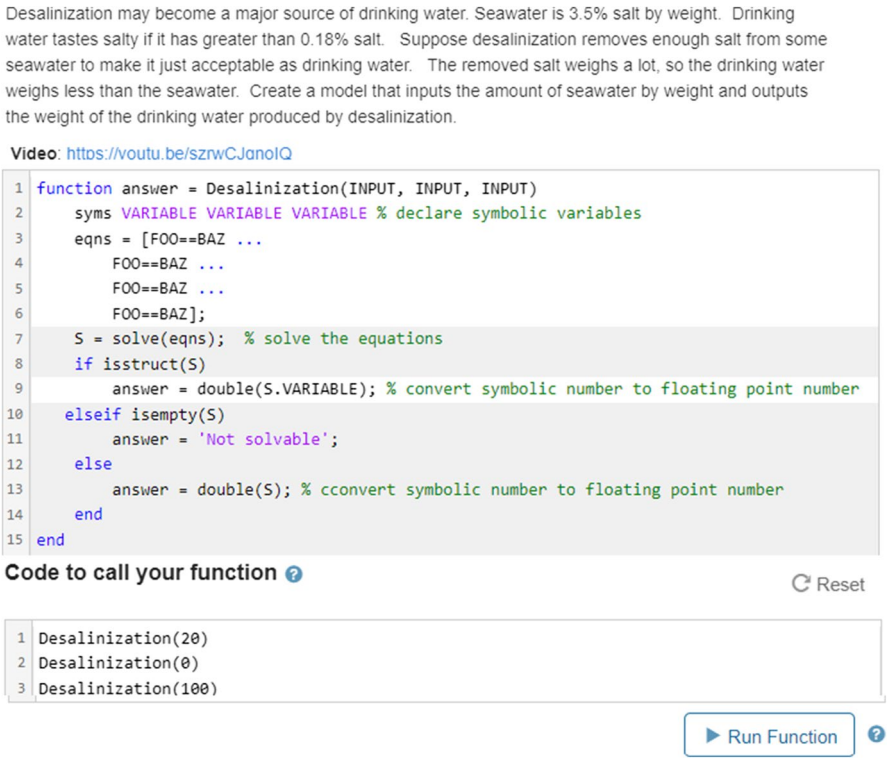

Fig. 7 A model construction problem in Matlab Grader

```
function answer = Desalinization(SeaWater)
    syms SeaSalt DrinkingSalt DrinkingWater PureWater % declare symbolic variables
    eqns = [SeaSalt == SeaWater * 0.035 ...
        DrinkingSalt == DrinkingWater * 0.0018 ...
        SeaWater == SeaSalt + PureWater ...
        DrinkingWater == DrinkingSalt + PureWater];
    S = solve(eqns);  % solve the equations
    if isstruct(S)
        answer = double(S.DrinkingWater); % convert symbolic number to floating point number
   elseif isempty(S)
        answer = 'Not solvable';
    else
        answer = double(S); % cconvert symbolic number to floating point number
    end
end
```

Fig. 8 Solution to the problem of Fig. 7

- The model should only mention numbers in the problem statement
- Every equation should have just one type of operator: +, -, *, / or ^. Repetitions of + and * are OK, so Foo == Baz + Zorch + Mumble is OK.
- Variables should have meaningful names. In particular, students should not use x, y or z.

Students were told that points would be taken off during exams and quizzes if they violated these rules. During homework and practice quizzes, students could submit problems as many times as they wanted in order to get input–output feedback. During quizzes and exams, there were no test cases. Instead, their solutions were manually scored and the 3 rules above were enforced.

Students had already used Matlab for several weeks, so they were somewhat familiar with the programing language. They were taught two new syntactic conventions for entering equations: (1) Use "==" instead of "=" (2) Use "…" after all equations except the last one.

The content mirrored the OMRaaT content exactly. The same lectures, examples and problems were used. Most importantly, the instruction taught the same 4 schemas in the same order, with the same focus on decomposing a problem into atomic relationships and on sharing variables between schema applications.

In summary, the MATLAB Grader instruction differed in four ways from the OMRaaT instruction: (1) The crucial difference was that MATLAB Grader students got only answer-based feedback instead of step-based feedback. That is, they had to enter a whole model in order to get feedback on its correctness. In contrast, OMRaaT gave feedback on the steps required for constructing a model. (2) Because some MATLAB Grader students needed more than answer-based feedback, every problem had a video of the instructor solving the problem. On average, each video received 0.44 views per student. (3) As explained earlier, the models had to have inputs and outputs, so the problems were reworded. (4) The OMRaaT students learned about about human problem solving as they learned algebraic model construction, whereas the MATLAB Grader students learned about algebraic modeling only.

The MATLAB Grader tutoring system is typical of many tutoring systems that give answer-based feedback. The feedback is often just binary: correct vs. incorrect. They often provide help via an instructor video.

The Study

The regression discontinuity study of OMRAaaT, which was described in the preceding section, was conducted in fall 2019 in a junior-level class on modeling. The MATLAB Grader tutoring system was used the following year in the same class. The study described here compared learning gains of 2019 students using OMRaaT to learning gains of 2020 students using the MATLAB Grader tutoring system.

The same tests and scoring procedure were used in both years. The tests were given on the Solver in 2019 and on MATLAB Grader in 2020, so the MATLAB Grader problems had inputs and outputs but the Solver problems did not. As mentioned earlier, there were 3 problems on the 2019 pretest where the student scores were at ceiling so these problems were excluded when scoring the 2019 pretest and mid-test. These 3 problems were not included on the 2020 pretest and posttests.

In the regression discontinuity study, OMRaaT was used only by students whose pretest score was below the pretest median, 11.5. In the following year, the MATLAB Grader tutoring system was used by all students. In order to have a fair comparison, this study included only students who scored below 11.5 on the MATLAB Grader pretest. Both the OMRaaT group and the MATLAB Grader group had only consenting students who did both tests. The OMRaaT group had 23 students, and the MATLAB Grader group had 42 students.

Results

Figure 9 shows the mean pretest, posttest and gains scores for both groups. In this discussion, the 2019 mid-test is called "posttest". The pretest means were not reliably different ($p = 0.988$). The posttest means were reliably different ($p = 0.003$, two-tailed T-test), as were the means of the gain scores ($p < 0.001$, two tailed T-test). The effect size (d) was 0.84. The OMRaaT students' gains were approximately double the gains of the MatLab Grader students. The step-based tutoring of OMRaaT appears to be more effective than the answer-based tutoring of the MATLAB Grader tutoring system.

It could be that interpreting answer-based feedback requires more effort and skill than interpreting step-based feedback. If so, then the results might show an aptitude-treatment interaction. That is, the larger the pretest score, the larger the gain among students using the MATLAB Grader tutoring system. In order to check for this effect, Fig. 10 shows a scatterplot of both groups. Regression lines are also plotted. Because the slopes of the regression lines are both close to 1.0, there appears to be no aptitude-treatment interaction in either group.

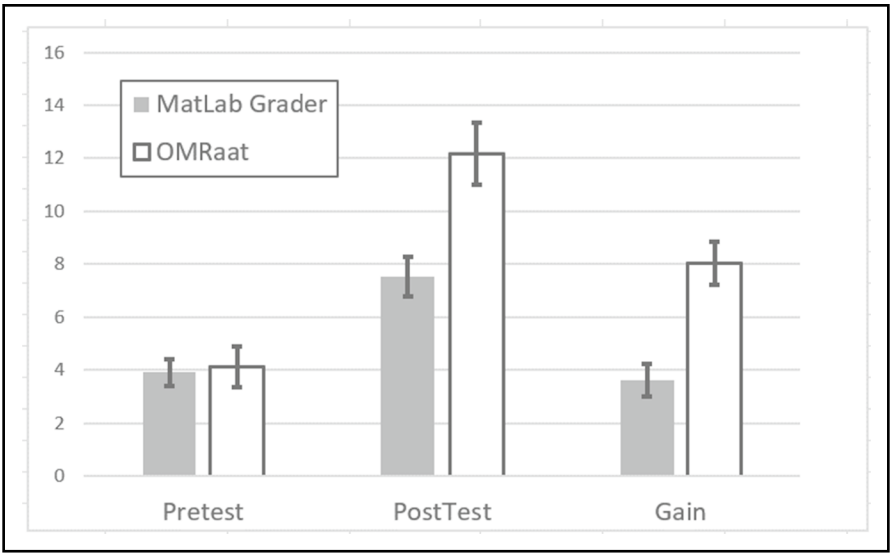

Fig. 9 Means and standard errors of the mean

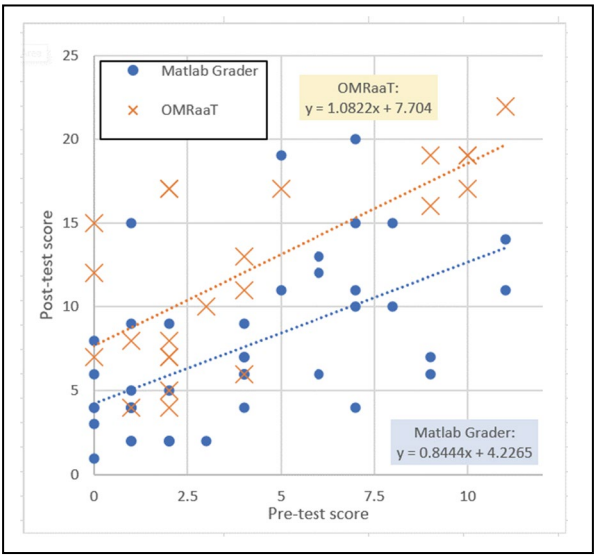

Fig. 10 Scatterplots of both treatments

Discussion

The bottom line is that there appears to be a reliable ($p = 0.023$) moderately large ($d = 0.67$) positive effect when below-median OMRaaT students were compared to

above-median students who received no instruction. Also, when students below the pretest median used either OMRaaT or an answer-based tutoring system, there was a reliable ($p < 0.001$) large ($d = 0.84$) advantage in learning gains for using OMRaaT.

The fact that the OMRaaT instruction improves the scores of underachieving students is remarkable and welcome. It is also welcome to see more data consistent with the hypothesis that step-based tutoring (OMRaaT) is more effective than answer-based tutoring (the MATLAB Grader tutoring system), as discussed by VanLehn (2011).

However, it is too early to declare victory over the notorious educational barrier of algebra word problems. From Fig. 10, we can see that there is a cluster of seven OMRaaT students who scored less than 9 on the mid-test. These students benefited from OMRaaT, but they did not benefit enough. These students appear to need more help. That is a challenge for future work.

Concerning transfer of model-construction skill from school/university to real-life settings, we point out that failure in school almost always results in failure to real life. The OMRaaT project focusses on students who failed to learn how to construct models during their high school and introductory undergraduate mathematics classes. If we can get some of those failing students to learn how to solve school problems, that at least allows some hope that their new skill will transfer to real-life settings.

For technologists, several lessons can be learned from the OMRaaT project. The overall instruction is classic two-loop tutoring system (VanLehn, 2006, 2011). The outer loop executes once per problem. The inner loop executes once per step. The inner loop gives students feedback on each step. OMRaaT itself only handles the inner (step) loop of tutoring (VanLehn, 2006). Canvas implements the outer loop. OMRaaT is an example-tracing tutoring system (Aleven et al., 2009) in that it does not generate a solution to a problem but instead utlilizes a solution given to it by the problem's author. Due to this simple design and its mundane user interface, the implementation of OMRaaT took only 4 months of full-time work by an experienced JavaScript programmer using the React framework. The lesson to be learned is that even if one doesn't use an authoring tool like CTAT (Aleven et al., 2015), it doesn't always take long to implement the inner loop of a step-based tutoring system if it is example-tracing and has a simple user interface.

The second technical lesson is that we were able to use a conventional LMS, Canvas, to implement the outer loop. An assignment in Canvas usually contains multiple problems, examples, etc. We used each assignment to present just one problem or example. Having 47 assignments cluttered the Canvas gradebook, but also allowed the gradebook to be used as a dashboard by the instructor to detect students who had fallen behind.

We are encouraged by the results so far, so we plan to try to make the OMRaaT module even more effective. The first task will be to analyze data from treatment students who did not gain much. One possibility is that the module was paced too quickly for the non-gainers; they may need even more examples and problems than they were given. SBI instruction (Fuchs et al., 2003, 2004a, b, 2009, 2010; Jitendra et al., 2007, 2009, 2011, 2013, 2015; Xin et al., 2005) usually lasted for 16 weeks, albeit with younger students. A second possibility is that the non-gainers figured out

ways to game OMRaaT (Baker et al., 2004) so that they could make rapid progress and yet still not learn much. There are many other possible explanations as well. Regardless of what happened to cause a few students to gain little, we are pleased to see that most students in the underachieving treatment group had large learning gains.

Funding This research was supported by NSF 1628782, NSF 1840051 and The Diane and Gary Tooker Chair for Effective Education in Science, Technology, Engineering and Math.

Declarations

Conflict Interests The authors have no relevant financial or non-financial interests to disclose, nor conflicting interests nor competing interests.

References

Aleven, V., McLaren, B., Sewall, J., & Koedinger, K. R. (2009). A new paradigm for intelligent tutoring systems: Example-tracing tutors. *International Journal of Artificial Intelligence in Education, 19*, 105–154.

Aleven, V., Sewall, J., Popescu, O., van Velsen, M., Demi, S., & Leber, B. (2015). Reflecting on twelve years of ITS authoring tools research with CTAT. In R. Sottilare, A. C. Graesser, X. Hu, & K. Brawner (Eds.), *Design Recommendations for Adaptive Intelligent Tutoring Systems (Vol. III, Authoring Tools)* (pp. 263–283). US Army Research Laboratory.

Anthony, L., Corbett, A. T., Wagner, A. Z., Stevens, S. M., & Koedinger, K. R. (2004). Student question-asking patterns in an intelligent algebra tutor. In J. C. Lester, R. M. Vicari, & F. Praguacu (Eds.), *Intelligent Tutoring Systems: 7th International Conference, ITS 2004* (pp. 455–467). Springer-Verlag.

Arnau, D., Arevalillo-Herraez, M., Puig, L., & Gonzalez-Calero, J. A. (2013). Fundamentals fo the design and the operation of an intelligent tutoring system for the learning of the arithmetical and algebraic way of solving word problems. *Computer and Education, 63*, 119–130.

Arroyo, I. (2000). AnimalWatch: An arithmetic ITS for elementary and middle school students. In *Presented at the Workshop at ITS 2000*.

Avouris, N., Margaritis, M., Komis, V., Saez, A., & Melendez, R. (2003). ModellingSpace: Interaction design and architecture of a collaborative modelling environment. In *Presented at the Sixth International Conference on Computer Based Learning in Sciences (CBLIS)*.

Baker, R. S. J. s., Corbett, A., Koedinger, K. R., & Wagner, A. Z. (2004). Off-task behavior in the cognitive tutor classroom: When students "Game the System". In E. Dykstra-Erickson & M. Tscheligi (Eds.), *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 383–390)* ACM.

Beal, C., Arroyo, I., Cohen, P. R., & Woolf, B. P. (2010). Evaluation of AnimalWatch: An intelligent tutoring system for arithmetic and fractions. *Journal of Interactive Online Learning, 9*(1), 64–77.

Beck, J., Woolf, B. P., & Beal, C. (2000). *ADVISOR: A machine learning architecture for intelligent tutor construction* (pp. 552–557). CA, AAAIPress.

Beek, W., Bredeweg, B., & Lautour, S. (2011). Context-dependent help for the DynaLearn modelling and simulation workbench. In G. Biswas (Ed.), *Artificial Intelligence in Education* (pp. 4200–4422). Springer-Verlag.

Biswas, G., Leelawong, K., Schwartz, D. L., & Vye, N. J. (2005). Learning by teaching: A new agent paradigm for educational software. *Applied Artificial Intelligence, 19*, 263–392.

Blessing, S. B., & Ross, B. H. (1996). Content effects in problem categorization and problem solving. *Journal of Experimental Psychology: Learning, Memory and Cognition, 22*(3), 792–810.

Bravo, C., van Joolingen, W. R., & de Jong, T. (2009). Using Co-Lab to build system dynamics models: Students' actions and on-line tutorial advice. *Computer and Education, 53*, 243–251.

Bredeweg, B., & Forbus, K. D. (2003). Qualitative modeling in education. *AI Magazine, 24*(4), 35–46.

Bredeweg, B., Liem, J., Beek, W., Salles, P., & Linnebank, F. (2010). Learning spaces as representational scaffolds for learning conceptual knowledge of system behavior. In M. Wolpers (Ed.), *EC-TEL* (pp. 46–61). Springer-Verlag.

Bridewell, W., Sanchez, J. N., Langley, P., & Billman, D. (2006). An interactive environment for the modeling and discovery of scientific knowledge. *International Journal of Human-Computer Studies, 64*, 1099–1114.

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.

Chang, K.-E., Sung, Y.-T., & Lin, S.-F. (2006). Computer-assisted learning for mathematical problem solving. *Computers & Education, 46*, 140–151.

Chase, C. C., Chin, D. B., Oppenzzo, M., & Schwartz, D. L. (2009). Teachable agents and the Protégé effect: Increasing the effort towards learning. *Journal of Science Education and Technology, 18*(4), 334–352.

Chi, M., & VanLehn, K. (2008). Eliminating the gap between the high and low students through meta-cognitive strategy instruction. In B. P. Woolf, E. Aimeur, R. Nkambou, & S. P. Lajoie (Eds.), *Intelligent Tutoring Systems: 9th International Conference: ITS2008* (pp. 603–613). Springer.

Chi, M., & VanLehn, K. (2010). Meta-cognitive strategy instruction in intelligent tutoring systems: How, when and why. *Journal of Educational Technology and Society, 13*(1), 25–39.

Connelly, J., & Katz, S. (2009). Toward more robust learning of physics via reflective dialogue extensions. In G. Siemens & C. Fulford (Eds.), *Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications 2009* (pp. 1946–1951). AACE.

Cook, J. L. (2006). College students and algebra story problems: Strategies for identifying relevant information. *Reading Psychology, 27*, 95–125.

Cooper, G., & Sweller, J. (1987). Effects of schema acquisition and rule automation on mathematical problem-solving transfer. *Journal of Educational Psychology, 79*(4), 347–362.

Corbett, A., Wagner, A. Z., & Raspat, J. (2003). The impact of analysing example solutions on problem solving in a pre-algebra tutor. In U. Hoppe, F. Verdejo, & H. Kay (Eds.), *Artificial Intelligence in Education: Proceedings of AIED 2003: The 11th International conference on AI in Education* (pp. 133–140). IOS Press.

Corbett, A., Wagner, A. Z., Chao, C.-Y., Lesgold, S., Stevens, S. M., & Ulrich, H. (2005). Student questions in a classroom evaluation of the ALPS learning environment. In C.-K. Looi & G. McCalla (Eds.), *Artificial Intelligence in Education* (pp. 780–782). IOS Press.

Corbett, A., Wagner, A. Z., Lesgold, S., Ulrich, H., & Stevens, S. M. (2006). The impact of learning of generating vs. selecting descriptions in analyzing algebra example solutions. In S. A. Barab, K. E. Hay, & D. T. Hickey (Eds.), *The 7th International Conference of the Learning Sciences* (pp. 99–105). Erlbaum.

Cummins, D. D., Kintsch, W., Reusser, K., & Weimer, R. (1988). The role of understanding in solving word problems. *Cognitive Psychology, 20*, 405–438.

de Jong, T., Martin, E., Zamarro, J.-M., Esquembre, F., Swaak, J., & van Joolingen, W. R. (1999). The integration of computer simulation and learning support: An example from the physics domain of collisions. *Journal of Research in Science Teaching, 36*(5), 597–615.

Derry, S. J., & Hawkes, L. W. (1993). Local cognitive modeling of problem-solving behavior: An application of Fuzzy Theory. In S. P. Lajoie & S. J. Derry (Eds.), *Computers as Cognitive Tools* (pp. 107–140). Lawrence Erlbaum Associates.

Forbus, K. D., Carney, K., Sherin, B. L., & Ureel II, L. C. (2005). VModel: A visual qualitative modeling environment for middle-school students. *AI Magazine, 26*(3), 63–72.

Fuchs, L. S., et al. (2003). Explicitly teaching for transfer: Effects on third-grade students' mathematical problem solving. *Journal of Educational Psychology, 95*(2), 293–305.

Fuchs, L. S., Fuchs, D., Finelli, R., Courey, S. J., & Hamlett, C. L. (2004a). Expanding schema-based transfer instruction to help third graders solve real-life mathematical problems. *American Education Research Journal, 41*(2), 419–445.

Fuchs, L. S., Fuchs, D., Prentice, K., Hamlett, C. L., Finelli, R., & Courey, S. J. (2004b). Enhancing mathematical problem solving among third-grade students with schema-based instruction. *Journal of Educational Psychology, 96*(4), 635–647.

Fuchs, L. S., et al. (2009). Remediating number combinations and word problem deficits among students with mathematics difficulties: A randomized control trial. *Journal of Educational Psychology, 101*(3), 561–576.

Fuchs, L. S., et al. (2010). The effects of schema-broadening instruction on second grader's word-problem performance and their ability to represent word problems with algebric equations: A randomized control study. *The Elementary School Journal, 110*(4), 440–463.

Fuchs, L. S., Fuchs, D., Seethaler, P. M., & Barnes, M. A. (2019). Addressing the role of working memory in mathematical word-problem solving when designing intervention for struggling learners. *ZDM Mathematics Education, 52*, 87–96.

Gerjets, P., Scheiter, K., & Catrambone, R. (2004). Desiging instructional examples to reduce intrinsic cognitive load: Molar versus modular presentation of solution procedures. *Instructional Science, 32*, 33–58.

Gerjets, P., Scheiter, K., & Catrambone, R. (2006). Can learning from molar and modular worked examples be enhanced by providing instructional explanations and prompting self-explanations? *Learning and Instruction, 16*, 104–121.

Gould, L., & Finzer, W. (1982). A study of TRIP: A computer system for animating time-rate-distance problems. *International Journal of Man-Machine Studies, 17*, 109–126.

Heffernan, N. T. (2003). Web-based evaluations showing both cognitive and motivational benefits of the Ms. Lindquist tutor. *Proceedings of the 11th International Conference on Artificial Intelligence in Education.* Berlin, Springer-Verlag.

Heffernan, N. T., & Croteau, E. A. (2004). Web-based evaluations showing differential learning for tutorial strategies employed by Ms. Lindquist tutor. In J. C. Lester, R. M. Vicari, & F. Parguaca (Eds.), *Intelligent Tutoring Systems: 7th International Conference, ITS 2004* (pp. 491–500). Springer-Verlag.

Heffernan, N. T., & Koedinger, K. R. (1997). The composition effect in symbolizing: The role of symbol production vs. text comprehension. In M. G. Shafto & P. Langley (Eds.), *Proceedings of the Ninetheenth Annual Meeting of the Cognitive Science Society* (pp. 307–312). Erlbaum.

Heffernan, N. T., Koedinger, K. R., & Razzaq, L. (2008). Expanding the model-tracing architecture: A 3rd generation intelligent tutor for algebra symbolization. *International Journal of Artificial Intelligence in Education, 18*, 153–178.

Hutchinson, N. L. (1993). Effects of cognitive strategy instruction on algebra problem solving of adolescents with learning disabilities. *Learning Disability Quarterly, 16*, 34–63.

Jitendra, A. K., Griffin, C. C., Haria, P., Leh, J., Adams, A., & Kaduvettoor, A. (2007). A comparison of single and multiple strategy instruction on third-grade students' mathematical problem solving. *Journal of Educational Psychology, 99*(1), 115–127.

Jitendra, A. K., et al. (2009). Improving seventh grade students' learning of ratio and proportion: The role of schema-based instruction. *Contemporary Educational Psychology, 34*, 250–264.

Jitendra, A. K., Star, J. R., Rodriguez, M., Lindell, M., & Someki, F. (2011). Improving students' proportional thinking using schema-based instruction. *Learning and Instruction, 21*, 731–745.

Jitendra, A. K., Star, J. R., Dupuis, D. N., & Rodiguez, M. C. (2013). Effectiveness of schema-based instruction for improving seventh-grades students' proportional reasoning: A randomized experiment. *Journal of Research on Educational Effectiveness, 6*(2), 114–136.

Jitendra, A. K., et al. (2015). Effects of a research-based intervention to improve seventh-grade students' proportional problem solving: A cluster randomized trial. *Journal of Educational Psychology, 107*(4), 1019–1034.

Katz, S., Allbritton, D., & Connelly, J. (2003). Going beyond the problem given: How human tutors use post-solution discussions to support transfer. *International Journal of Artificial Intelligence in Education, 13*, 79–116.

Katz, S., Connelly, J., & Wilson, C. (2007). Out of the lab and into the classroom: An evaluation of reflective dialogue in Andes. In R. Luckin & K. R. Koedinger (Eds.), *Proceedings of AI in Education, 2007* (pp. 425–432). IOS Press.

Kintsch, W., & Greeno, J. G. (1985). Understanding and solving word arithmetic problems. *Psychological Review, 92*, 109–129.

Koedinger, K. R., & Anderson, J. R. (1998). Illustrating principled design: The early evolution of a cognitive tutor for algebra symbolization. *Interactive Learning Environments, 5*, 161–180.

Koedinger, K. R., & Nathan, M. J. (2004). The real story behind story problems: Effects of representations on quantitative reasoning. *Journal of the Learning Sciences, 13*(2), 129–164.

Koedinger, K. R., Alibali, M. W., & Nathan, M. J. (2008). Trade-offs between grounded and abstract representations: Evidence from algebra problem solving. *Cognitive Science, 32*, 366–397.

Kurtz dos Santos, A., & d. C., & Ogborn, J. (1994). Sixth form students' ability to engage in computational modelling. *Journal of Computer Assisted Learning, 10*(3), 182–200.

Leelawong, K., & Biswas, G. (2008). Designing learning by teaching agents: The Betty's Brain system. *International Journal of Artificial Intelligence in Education, 18*(3), 181–208.

Lipsey, M. W., Puzio, K., Yun, C., Hebert, M. A., Steinka-Fry, K., Cole, M. W., Roberts, M., Anthrony, K. S., & Busick, M. E. (2012). *Translating The Statistical Representation of the Effects of Education Interventions into More Readily Interpretable Forms.* IES: National Center for Special Education Research. US Department of Education.

Löhner, S. (2005). *Computer based modeling tasks: The role of external representation. Ph. D*. Faculty of Social and Behavioural Sciences, University of Amsterdam.

Löhner, S., Van Joolingen, W. R., & Savelsbergh, E. R. (2003). The effect of external representation on constructing computer models of complex phenomena. *Instructional Science, 31*, 395–418.

Löhner, S., Van Joolingen, W. R., Savelsbergh, E. R., & Van Hout-Wolters, B. (2005). Students' reasoning during modeling in an inquiry learning environment. *Computers in Human Behavior, 21*, 441–461.

Looi, C.-K., & Tan, B. T. (1996). WORDMATH: A computer-based environment for learning word problem solving. In *Presented at the Conmputer Aided Learning and Instruction in Science and Engineering*. Springer.

Looi, C.-K., & Tan, B. T. (1998). A cognitive apprenticeship-based environment for learning word problem solving. *Journal of Computers in Mathematics and Science Teaching, 17*(4), 339–354.

Marshall, S. P. (1995). *Schemas in problem solving*. Cambridge University Press.

Marshall, S. P., Barthuli, K. E., Brewer, M. A., & Rose, F. E. (1989). *Story problem solver: A schema-based system of instruction*. San Diego State University.

McArthur, D., Lewis, M., Ormseth, T., Robyn, A., Stasz, C., & Voreck, D. (1989). *Algebraic thinking tools: Support for modeling situations and solving problems in Kids' World*. RAND Corporation.

Metcalf, S. J. (1999). *The design of guided learning-adaptable scaffolding in interactive learning environments. Ph. D., Computer Science and Engineering, University of Michigan*. Ann Arbor, MI.

Metcalf, S. J., Krajcik, J., & Soloway, E. (2000). Model-it: A design retrospective. In M. J. Jacobson & R. B. Kozma (Eds.), *Innovations in science and mathematics education: Advanced designs for technologies of learning* (pp. 77–115). Routledge.

Mulder, Y. G., Lazonder, A., & de Jong, T. (2010). Finding out how they find it out: An empirical analysis of inquiry learners' need for support. *International Journal of Science Learning, 32*(15), 2033–2053.

Mulder, Y. G., Lazonder, A. W., de Jong, T., Anjewierden, A., & Bollen, L. (2011). Validating and optimizing the effects of model progression in simulation-based inquiry learning. *Journal of Science Education and Technology, 21*, 722–729.

Munez, D., Orrantia, J., & Rosales, J. (2013). The effect of external representations on compare word problems: Supporting mental model construction. *Journal of Experimental Education, 81*(3), 337–355.

Nathan, M. J., Kintsch, W., & Young, E. (1992). A theory of algebra-word-problem comprehension and its implications for the design of learning enviroments. *Cognition and Instruction, 9*(4), 329–389.

NGA & CCSSO. (2011). Common Core Sate Standards for Mathematics. Downloaded from https://www.corestandards.org on October 31, 2011.

NGSS. (2013). *Next generation science standards: For states, by states*. The National Academies.

Pareto, L., Arvemo, T., Dahl, Y., Haake, M., & Gulz, A. (2011). A teachable-agent arithmetic game's effects on mathematics understanding, attitude and self-efficacy. In G. Biswas & S. Bull (Eds.), *Proceedings of Artificial Intelligence in Education* (pp. 247–255). Springer.

Pauli, C., & Reusser, K. (1997). Supporting collaborative problem solving: Supporting collaboration and supporting problem solving. In *Presented at the Proceedings of Swiss Workshop on Collaborative and Distributed Systems*.

Quinn, J., & Alessi, S. M. (1994). The effects of simulation complexity and hypothesis-generation strategy on learning. *Journal of Research in Computing in Education, 27*(1), 75–92.

Ramachandran, S. (2003). A meta-cognitive computer-based tutor for high-school algebra. In D. Lassner & C. McNaught (Eds.), *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2003* (pp. 911–914). AACE.

Reif, F., & Scott, L. A. (1999). Teaching scientific thinking skills: Students and computers coaching each other. *American Journal of Physics, 67*(9), 819–831.

Reimann, P. (2011). Design-based research. In L. Markauskaite, P. Freebody, & J. Irwin (Eds.), *Methodological choice and design: Scholarship, policy and practice in social and educational research* (pp. 37–50). Springer.

Renkl, A., Stark, R., Gruber, H., & Mandl, H. (1998). Learning from worked-out examples: The effects of example variability and elicited self-explanations. *Contemporary Educational Psychology, 23*, 90–108.

Reusser, K. (1993). Tutoring systems and pedagogical theory: Representational tools for understanding, planning and reflection in problem solving. In S. P. Lajoie & S. J. Derry (Eds.), *Computers as Cognitive Tools* (pp. 143–178). Lawrence Erlbaum Associates.

Riley, M. S., & Greeno, J. G. (1988). Developmental analysis of understanding language about quantities and of solving problems. *Cognition and Instruction, 5*(1), 49–101.

Schwartz, D. L., et al. (2009). Interactive metacognition: Monitoring and regulating a teachable agent. In D. J. Hacker, J. Dunlosky, & A. C. Graesser (Eds.), *Handbook of Metacognition in Education* (pp. 340–358). Taylor & Francis.

Segedy, J. R., Kinnebrew, J. S., & Biswas, G. (2012). Supporting student learning using conversational agents in a teachable agent environment. In *Presented at the Proceedings of the 10th International Conference of the Learning Sciences*. Australia.

Swaak, J., van Joolingen, W. R., & de Jong, T. (1998). Supporting simulation-based learning; The effects of model progression and assignments on definition and intuitive knowledge. *Learning and Instruction, 8*(3), 235–252.

Sweller, J., Mawer, R. F., & Ward, M. R. (1983). Development of expertise in mathematical problem solving. *Journal of Experimental Psychology: General, 112*, 629–661.

van Joolingen, W. R., De Jong, T., Lazonder, A., Savelsbergh, E. R., & Manlove, S. (2005). Co-Lab: Research and development of an online learning environment for collaborative scientific discovery learning. *Computers in Human Behavior, 21*, 671–688.

VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education, 16*, 227–265.

VanLehn, K. (2008). The Interaction Plateau: Answer-based tutoring < step-based tutoring = natual tutoring (abstract only). In B. P. Woolf, E. Aimeur, R. Nkambou, & S. P. Lajoie (Eds.), *Intelligent Tutoring systems 2008* (p. 7). Springer-Verlag.

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems and other tutoring systems. *Educational Psychologist, 46*(4), 197–221.

VanLehn, K. (2013). Model construction as a learning activity: A design space and review. *Interactive Learning Environments, 21*(4), 371–413.

Vanlehn, K., Banerjee, C., Milner, F., & Wetzel, J. (2020). Teaching algebraic model construction: A tutoring system, lessons learned and an evaluation. *International Journal of Artificial Intelligence in Education, 30*(3), 459–480.

White, B. Y. (1984). Designing computer games to help physics students understand Newton's Laws of Motion. *Cognition and Instruction, 1*(1), 69–108.

White, B. Y. (1993). ThinkerTools: Causal models, conceptual change and science education. *Cognition and Instruction, 10*(1), 1–100.

White, B. Y., & Frederiksen, J. R. (1990). Causal model progressions as a foundation for intelligent learning environments. *Artificial Intelligence, 42*, 99–157.

Willis, G. B., & Fuson, K. C. (1988). Teaching children to use schematic drawings to solve addition and subtraction word problems. *Journal of Educational Psychology, 80*(2), 192–201.

Xin, Y. P., Zhang, D., Park, J. Y., Tom, K., Whipple, A., & Si, L. (2001). A comparison of two mathematics problem-solving strategies: Facilitate algebra-readiness. *The Journal of Educational Research, 104*(6), 381–395.

Xin, Y. P., Jitendra, A. K., & Deatline-Buchman, A. (2005). Effects of mathematical word problem-solving instruction on middle school students with learning problems. *The Journal of Special Education, 39*(3), 181–192.

Zhang, L., et al. (2014). Evaluation of a meta-tutor for constructing models of dynamic systems. *Computers & Education, 75*, 196–217.