Int J Artif Intell Educ (2014) 24:216–250
DOI 10.1007/s40593-014-0016-x

RESEARCH ARTICLE

Reinforcing Math Knowledge by Immersing Students in a Simulated Learning-By-Teaching Experience

Douglas B. Lenat · Paula J. Durlach

Published online: 21 May 2014

Abstract We often understand something only after we've had to teach or explain it to someone else. Learning-by-teaching (LBT) systems exploit this phenomenon by playing the role of *tutee*. BELLA, our sixth-grade mathematics LBT systems, departs from other LTB systems in several ways: (1) It was built not from scratch but by very slightly extending the ontology and knowledge base of an existing large AI system, Cyc. (2) The "teachable agent"—Elle—begins not with a *tabula rasa* but rather with an understanding of the domain content which is close to the human student's. (3) Most importantly, Elle never actually learns anything directly from the human tutor! Instead, there is a super-agent (Cyc) which already knows the domain content extremely well. BELLA builds up a mental model of the human student by observing them interact with Elle. It uses that Socratically to decide what Elle's current mental model should be (what concepts and skills Elle should already know, and what sorts of mistakes it should make) so as to best help the user to overcome *their* current confusions. All changes to the Elle model are made by BELLA, not by the user—the only *learning* going on is BELLA learning more about the user—but from the user's point of view it often *appears* as though Elle were attending to them and learning from them. Our main hypothesis is that this may prove to be a particularly powerful and effective illusion to maintain.

Keywords Intelligent tutoring systems · Learning-by-teaching · Mathematics · Student model · Software agent · Teachable agents

This paper is being submitted in the category of "*New Ideas* that could lead to transformative educational technology in mathematics and science". A Special Issue on Landmark Learning Systems and New Ideas and Developments in Mathematics and Science Learning in memory of Erica Melis

D. B. Lenat (✉)
Cycorp, Austin, TX, USA
e-mail: lenat@cyc.com

P. J. Durlach
Advanced Distributed Learning Initiative, Orlando, FL, USA
e-mail: paula.durlach@adlnet.gov

Introduction

We are in the process of applying Cyc, an enormous symbolic Artificial Intelligence platform (Lenat et al. 1985, 2010; Lenat and Guha 1990), to function as an ITS (intelligent tutoring system) for 6th grade mathematics. *Applying* Cyc means slightly extending its ontology of 500,000 terms, its knowledge base of ten million assertions and rules, its representation language (first order predicate calculus with several second- and higher-order extensions), its inference engine (which comprises a general theorem prover and 1,100 special-purpose efficient reasoners), and its natural language generation/explanation subsystem. In addition to making these five relatively small extensions to Cyc, we have designed and developed a substantial new gamification element and corresponding interface. This extended version of Cyc has its own name: BELLA.

Most ITS's simulate a teacher; the user plays the role of the student (Fletcher 2011). *Teachable agent* systems, which are surveyed in the following section, are ITS's which reverse this paradigm and simulate a learning agent, which the user is then placed into the role of teaching. These are also called learning-by-teaching (LBT) systems. BELLA appears to the user to be that sort of teachable agent system, though in actuality it is something rather different: BELLA already understands 6th grade math content, it merely runs an avatar, called Elle, who appears to need help in that subject, who has an even shakier understanding of pre-algebra than does the user. As the user—playing the role of tutor—interacts with Elle, BELLA (not Elle) watches and learns more and more about the user every time he/she corrects Elle, and even by his/her silence and inaction. That user model helps BELLA infer pedagogically useful changes to make in Elle's mental model, which then affect Elle's behavior on future problems. From the user's point of view, it often *appears* that Elle is being taught by them.

While we plan to experiment with various pedagogical principles for selecting the best Elle model for the user, at any given moment, initially the choice is to have Elle's model match the current user model as closely as possible with a slight exaggeration of each knowledge gap, incorrect skill, tendency to make a particular type of mistake or fall into a particular type of trap. For example, if the current user sometimes evaluates all operations left to right in an arithmetic expression, Elle will be given that same tendency with an even higher likelihood to the same sort of operator-precedence error, so Elle will make even more egregious and obvious such errors which the user will have a good chance of noticing, correcting, and hopefully, in the process of doing so, the user will refresh and improve their own understanding of operator precedence.

In cases where BELLA does not yet know some elements of the current user's model, it errs on the side of having Elle make those errors; if/when the user corrects Elle, two good things happen as a result: BELLA learns something it needed to know about this user and, at the same time, it has given the user a small positive esteem-building experience (esteem-building both in the very act of having caught the error and also in seeing that Elle is much less likely to make that error henceforth, thus believing that they have taught Elle something she needs to know.)

At some abstract level, BELLA truly is a LBT system, just like most teachable-agent systems, because its main function is to get the user to act like a tutor and, through that tutoring, to thereby deepen the user's own understanding of pre-algebra. But the process that is actually going on is not building up of "Elle's brain" by the user, rather it is dynamic diagnosis and treatment of the user by BELLA. Elle only *appears* to the user to be a teachable agent; Elle is just a puppet controlled by BELLA, a diagnostic instrument BELLA manipulates the way a doctor manipulates a stethoscope.

Before discussing BELLA in more detail, we review the origins of LBT and the current state of the art in LBT software systems. We then describe a cognitive task analysis we performed with several dozen sixth graders, and review relevant elements of Cyc. These three legs of the tripod (others' state-of-the-art work on learning-by-teaching systems; student observation; and Cyc) provide the context in which we then describe some of our design decisions, preliminary experiences and challenges, current efforts, and future plans for BELLA.

Learning-By-Teaching Systems

Seneca the Younger wrote more than 2,000 years ago, *docendo discimus*, i.e., *by teaching, we learn* (Stone 2004). Research has tended to confirm that teaching produces benefits for the teacher (e.g., Berliner 1989; Chase et al. 2009; Cohen et al. 1982; Michie et al. 1989; Roscoe and Chi 2007; Stern 2011). Biswas et al. (2001) suggest that three phases of teaching can enhance knowledge in the teacher: (1) planning, (2) explaining and (3) interpreting student questions and feedback. These can produce benefits in knowledge understanding for the teacher by promoting reflection, self-explanation, and studying for understanding (rather than memorizing). These, in turn, lead to better knowledge organization and schema structure. Learning-by-teaching has gone by several names, including LdL *(Lernen durch Lehren,* Graef and Preller 1994*)*, the Monitorial System (Tschurenev 2008), the Bell-Lancaster method (Lancaster 1821), learning companion systems (Uresti and du Boulay 2004), and teachable agent systems (e.g., Biswas et al. 2001; Chin et al. 2010; Matsuda et al. 2012a, b; Zhao et al. 2012).

Intelligent tutoring systems emerged in the 1920's in hard-wired mechanical form (Fry 1960), and transitioned to software in the 1960's, catalyzed by the development of the LOGO programming language (Feurzeig et al. 1969). For the last 50 years, most ITS's generally have had the computer "agent" playing the role of the teacher, and the student user playing the role of, well, the student. But there have been several ITS's which have experimented with the learning-by-teaching paradigm, generally calling themselves either "learning companion systems" (Uresti and du Boulay 2004) or "teachable agent systems" (Biswas et al. 2001; Chin et al. 2010; Matsuda et al. 2012a, b; Zhao et al. 2012).

For example, in Leelawong and Biswas (2008) and Segedy et al. (2013), a human student teaches "Betty"—a simulated tutee—about ecology by constructing a concept map. The map is made of nodes (ecological variables) and links representing causal relationships between nodes (causes increase or causes decrease). The user can ask Betty questions or have her take a generated quiz to determine how well she has learned. Inference rules applied to the map are used to generate Betty's answers. Feedback on quiz answers (provided by a mentor agent, Mr. Davis), ideally, prompt iterative cycles of reading and adding knowledge to or correcting the concept map, in order to enhance Betty's quiz performance. Other teachable agent systems have also used the building of concept maps, or other direct manipulation of the teachable agent's knowledge base as the technique by which the human student teaches a computer agent (Chin et al. 2010; Nichols 1994; Uresti and du Boulay 2004; Zhao et al. 2012). Research on Bettty's Brain (e.g., Wagster et al. 2007; Roscoe et al. 2013) has demonstrated that many students (acting as teachers) need some support for their metacognitive and self-regulated learning skills to fully reap the benefits of learning-by-teaching in these kinds of systems. This support includes such tactics as Betty or Mr. Davis asking the student to reflect on Betty's errors in order to focus on what she needs to learn, discouraging students from using trial and error methods for correcting the concept map, and Mr. Davis providing help when asked, in response to Betty's quiz performance. Adaptive prompts for reflection on what would best help a peer learner have also been found to be useful in peer tutoring (Walker et al. 2012). An attempt was made to extend this approach to 6th grade math (Husain et al. 2010; Katzlberger 2005), but the representation (a network of binary causes/impedes links) proved to be poorly impedance-matched to the complex concepts and procedural skills involved in this subject, unlike ecology (Blair et al. 2006; Brophy et al. 1999).

Several other teachable agent systems have also used the building of concept maps, or other direct manipulation of the teachable agent's knowledge base, as the technique by which the human student teaches a computer agent (e.g., Chin et al. 2010; Nichols 1994; Uresti and du Boulay 2004; Zhao et al. 2012). Generally the learning agent's model starts out empty, and is built up step by step by the user. This body of research has demonstrated that many human students (acting as teachers) need some support for their *meta*cognitive and self-regulated learning skills to fully reap the benefits of learning-by-teaching in these kinds of systems (Wagster et al. 2007; Roscoe et al. 2013).

Other digital LBT systems have more closely tried to mimic real teaching. Rather than the human tutor manipulating the teachable agent's "brain" directly, the human tutor does example problems and/or holds conversations with the agent. While not a dedicated teachable agent system, Operation ARIES! (Millis et al. 2011), has a component in which users can help a teachable agent better understand topics concerning scientific inquiry through conversation (e.g., independent vs. dependent variables). Trialogs can occur among the human user, a teacher agent, and a teachable agent, and can vary according to which participant plays the most active role as "teacher." These conversations are governed by a question, a pre-scripted ideal answer, and a set of hints and prompts based on expected partial answers or misconceptions, in a manner similar to AutoTutor (Graesser et al. 2005). When deemed to have sufficient knowledge, the human user in Operation ARIES! has the responsibility to "teach" the teachable agent, while the teacher agent mostly just observes.

Some LBT systems use artificial intelligence models of learning, generalization, and classification to learn (i.e., by induction, not by "being told.") In Winston (1970), users gave positive examples, negative examples, and near misses, to help the program learn the Boolean combination of features that define an "arch" composed of three blocks. Another early system, Math Concept Learning System (MCLS) (Michie et al. 1989), was designed to learn to solve linear equations in one variable. Its users could interact with the system in four ways: (1) They could show the system how to solve an equation by demonstration. (2) They could inspect the rules the system had already learned. (3) They could test the current rule they were trying to teach by asking the system to use that rule to solve an equation. (4) They could ask the system to solve an equation, with the user providing feedback for each step. Internally, MCLS represented problems as a set of attributes that could be either values or operators. MCLS learned how to solve equations by using past examples to induce the different actions that could be taken to solve equations (e.g., collect like terms) and when to apply each rule. SimStudent is a more recent version of the same general approach (e.g., Li et al. 2011; Matsuda et al. 2010, 2012a). SimStudent learns production rules by attempting to solve algebra equations and obtaining feedback from the human user on each step. If given negative feedback, SimStudent can try a different solution step or ask for a hint, in which case the user responds by performing the step. SimStudent occasionally asks users for explanations, such as why a particular problem was selected to work on, or why they provided negative feedback. The user responds to these requests by selecting a choice from a drop down menu or in free text. Like for the concept mapping systems, users can test their SimStudent by giving it a system-generated quiz. When quiz questions are fixed, Matsuda et al. (2012b), found that students may "teach to the quiz," resulting in a SimStudent that performs poorly on new problems. So, as with the concept net systems, students may apply strategies that are not conducive to deeper understanding (by the teachable agent and by themselves) unless meta-level mechanisms are built into the system to support their teaching.

In an effort to enhance student motivation and engagement, many digital learning-by-teaching systems include game-like elements (e.g., Chase et al. 2009; Kiili et al. 2012; Gulz et al. 2011; Matsuda et al. 2012a; Pareto et al. 2012; Zhao et al. 2012), and/or use animated characters to represent the teachable agents (e.g., Chin et al. 2010; Gulz et al. 2011; Kiili et al. 2012; Leelawong and Biswas 2008; Zhao et al. 2012). Many of these systems have a learning phase and a separate challenge phase, analogous to the quiz in Betty's Brain, but couched in more engaging activities For example, in one instantiation of SimStudent (Matsuda et al. 2012a), a game show-like competition between SimStudents was offered as an option (in addition to the quiz). Likewise, in the Eedu Elements math game (Kiili et al. 2012), human students instruct their teachable agents (mice) in a simulated classroom; but, subsequently each mouse must use its learning to compete with a cat in navigating a maze. The system developed by Gulz et al. (2011) and Pareto et al. (2012) more fully integrates the teaching and the game components. This system supports playing a card game requiring knowledge of base-ten concepts, such as place-value, carrying, and borrowing. The card game itself is played by two players, who could be a pair made up of any combination of human students, teachable agents, or system-generated agents. The human player can teach an agent how to play by (1) playing themselves, with the agent observing and asking questions, (2) allowing the teachable agent to play, and commenting on its actions, or (3) allowing the teachable agent to play unsupervised. Humans communicate with agents by selecting utterances from a menu of plausible response alternatives generated by the system. Pareto et al. (2012) found that when student pairs (9 and 10 year olds) were left to choose how they wanted to play, they adopted several different methods. For example many pairs team-supervised one teachable agent playing against a computer-generated agent; another frequently selected pattern was to collaborate on teaching two teachable agents as the agents played. On the whole, the self-selected patterns of play were more collaborative than competitive.

This paper presents an overview of a new LBT system currently under development and aimed at reinforcing mathematics knowledge in sixth graders (pre-algebra), a key foundation for future success in science, technology, engineering, and mathematics (STEM). Like some of the systems described above, the student—the user—interacts with an animated agent (named Elle) and helps her solve problems.

Our system, BELLA, extends the LBT paradigm, but departs from previous systems in several ways:

- BELLA "works" by building up and utilizing a symbolic knowledge model of the human student. Previous LBT systems, like SimStudent, support learning in a teachable agent and use the representation of what the teachable agent has learned to govern its problem solution behavior; however, they do not select the problems that the agent must solve based on an evaluation of what the human students themselves understand or misunderstand. For example, SimStudent allows the human student to choose the problems that SimStudent works on. In contrast, BELLA maintains a student model and uses that model to determine the nature of the next problem to be presented, in a manner analogous to many ITS's systems (VanLehn 2006) but unlike most LBT systems. In addition to influencing the nature of each problem, the student model also determines Elle's problem solving behavior and her interactions with the student. From the user's point of view, it seems as if they are teaching Elle; however, in actuality, Elle has multiple correct and incorrect parameterized models and the choice of models (and parameters) is made by BELLA based on the current user model. Thus, strictly speaking, Elle does not learn and is not a teachable agent. Rather, BELLA selects a model for Elle to follow, deemed to be pedagogically advantageous, based on the state of the student model.

  Our main hypothesis is that this very distinction—pretending to be taught versus "really" directly being taught by the user—may prove useful and powerful. This is precisely what enables BELLA in principle to be able to more wisely manipulate the user into better understanding the underlying subject matter than any "true LBT" system could. The latter must by necessity obey its user and believe its user,[1] starting as a *tabula rasa* and learning only what it has been told by its user. But BELLA is more like a parent who sees exactly what their child doesn't understand and then feigns ignorance very selectively, role-plays someone with an intelligently-selected set of foibles so as to maximally help their child see the error of his/her current misconceptions. BELLA is therefore better able (than true LBT systems) to make sure that the interaction between the human user and simulated tutee focuses on specific difficulties that this particular user suffers from. Similarly, BELLA is able to carefully reason about what the next math problem should be which this user faces, rather than leaving the training sequence up to the user (or totally fixed).

  In the coming year, we plan to gather rigorous, objective, and statistically significant data to test our central conjecture; as of yet we have only a small sample of anecdotal results (from sixth graders using BELLA) which are nonetheless interesting and encouraging; we discuss this later in this paper.

[1] Except in cases where the user tells the LBT system contradictory things, and it detects that.

- Most LBT systems have the teachable agent start out more or less as a blank slate, and the user incrementally builds up that agent model piece by piece. Elle starts with a mental model of pre-algebra which is almost at the same level as the user. If it has gaps or errors which the user lacks, those will get filled in quickly and give the user some feeling of accomplishment in the process. Elle's gaps and errors which mirror the user's may be exaggerated, to help the user notice them and repair them in Elle (and thence, in themselves). BELLA chooses what Elle to puppet, what face to pretend to show to the user. BELLA can also choose when to have Elle pretend to start relapsing, start forgetting previous lessons learned. In those hopefully rare cases where the user is hopelessly lost, too confused even to correct Elle, BELLA can decide to subtly—and briefly—reverse their tutor-tutee roles by having Elle appear to have an epiphany, a breakthrough which then allows her to explain her new insight to the user.
- The interactions all take place in the context of an engaging adventure game: Elle's spacecraft has crash-landed, and she faces numerous challenges that she requires help with (Fig. 3) Many intelligent tutoring systems—and many intelligent human teachers!—incorporate game elements to make the learning process more engaging (ATLT, 2012). But in BELLA, we have worked hard to make all the math problems emerge naturally in the course of that adventure, rather than being disembodied, isolated game show questions (e.g., Chase et al. 2009) or disembodied, isolated problems simply handed to the user (ATLT, 2012). For example, if Elle needs to add up several mixed numbers, it might be because she sees a map with segment distances marked, and she needs to find the shortest route to her destination, but not because "it was assigned". If the user skips the option to check their answer, because they are in a hurry, they might later have to backtrack and re-work the problem. If the user decides to guess at an answer or even bypass a problem entirely, they can do so, but over time they are led to see the consequences of that: that they have wasted time and resources by doing so, that time after time it's cost-effective to set things up as math problems and "do the math". The reader need look no further than Pokemon or Magic the Gathering or the "auction house" of almost any MMORPG to see overwhelming evidence that many of the same students who struggle with abstracted-out arithmetic and probability math problems in class are able to carry out more advanced versions of the same reasoning and skills in the context of a game.

Sixth Grade Math and Sixth Grade Math Students

STEM education is a key to technical competitiveness of individuals, organizations, and nations. And M is the foundation for S, T, and E. But mathematics is highly cumulative, and the knee of the curve appears to be pre-algebra (Brizuela et al. 2013). I.e., an individual's deep understanding of pre-algebra is a predictor of lifelong math and science literacy; and, conversely, a poor understanding of it is a predictor of lifelong STEM illiteracy and a bar to entering higher education. According to the President's Council of Advisors on Science and Technology (2010), about 70 % of U.

S. eighth graders scored below proficiency. So the present work targets what in the U.S. is 6th grade math: pre- algebra and pre-geometry.

In order to guide BELLA design decisions, we tested some of our initial design conceptions with sixth grade math students in Austin. In an initial session, 26 sixth graders were presented with four math word problems on the white board, and asked to complete them individually, showing their work. While the teacher was convinced these problems would pose no difficulty for her students, most of them made many errors and took significantly longer than she expected. Half the errors were not computational, but rather involved understanding the problem and solving the correct equations. In retrospect, the teacher believed that the results were due to her method of teaching: a blocked approach where practice problems in any given homework assignment or quiz are all of a very particular type, and for which students have just recently learned the step by step procedure for solving that type of problem. In contrast, our four problems were of multiple types, and required students to recall and apply procedures without the benefit of priming by recent instruction. ("We haven't had to know this stuff for *weeks*!") Their difficulty with this task highlights the need to reinforce prior learning and deepen conceptual understanding.

In another series of sessions, we observed peer tutoring on a new set of problems. We observed both naturalistic peer tutoring and peer tutoring in the form of a game we devised to parallel our initial BELLA user interface conception. In the game, one student takes on the role of the solver, and the other the clue-giver, and they collaborate to earn points by solving problems correctly, quickly, and with the fewest number of clues (Fig. 1).

The clue-giver had cards and tokens which they could place on the solver's worksheet or on the game board (with subtler clues costing less to the point score than more direct clues). Both situations suggested that sixth grade math students were not very good math tutors, even if they themselves were good at math. Rather than help the tutee, peer tutors had a tendency to just take over the problem and do it themselves. Peer tutors and clue-givers often provided inaccurate feedback, had difficulty providing explanations, and failed to recognize the possibility of alternative valid solution paths (for example, one tutor told their tutee they were wrong for multiplying two fractions' denominators before multiplying their numerators).

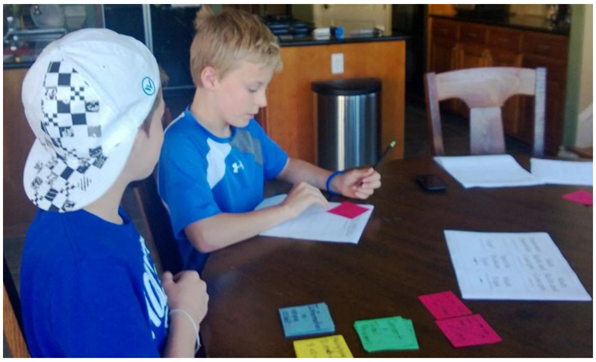


Fig. 1 A pair of students playing "Clued In", our paper-and-pencill version of BELLA. As the Solver works through a math problem, the Clue-Giver (tutor) selects cards and tokens to place directly on the Solver's worksheet or—for extra points—gives a more subtle hint by placing it on a *category* of error on the game board

We had much better results when the role of the tutee was played by a teacher or experimenter.

I.e., an adult played the role of the tutee and used their mental model of the student to determine what mistakes to make, when to appear stumped, and what questions to ask of the tutor. We found that students were eager to correct the adult's errors, and did so reliably with a bit of leading by the tutee. This observation led us to the position that BELLA might be more effective if users did not truly teach Elle, but only appeared to do so. Thus, rather than rely on the student teaching her, Elle is "taught" only by BELLA—i.e., whenever BELLA decides that this user will benefit from some change in Elle's model. Generally, BELLA keeps Elle functioning at a slightly lower level of comprehension than the user; i.e., Elle usually has an ablated version of the student model. As the student learns/regresses, BELLA improves/degrades Elle's mental model, so it will make fewer/more mistakes.

Even when adults role-played the tutee, students still had difficulty explaining why a step was correct or incorrect, however. This observation highlights the need to reinforce not just procedures, but also conceptual understanding (Roscoe and Chi 2007). BELLA addresses this in part by varying the way that Elle solves isomorphic problems, and in part by having Elle ask the tutor useful questions. Since Elle's mental model is only slightly lagging behind the user's, Elle can function more like a collaborator than a truly naïve tutee, and thus make leading suggestions and comments. Like SimStudent, Elle may sometimes request explanations; at other times, Elle may take self-corrective action and provide explanations especially when the student appears uncertain or stuck at a plateau.

Cyc and Bella

The Cyc artificial intelligence platform (Forbus 2012; Lenat and Guha 1990; Lenat et al. 1985, 2010) is the foundation of BELLA. Cyc comprises an ontology of half a million terms, a symbolic knowledge base (KB) of ten million hand-written rules and general pieces of information (e.g., water flows downhill; dogs have four legs). Cyc's KB contains a great deal of knowledge about *types* of events, places, professions, organizations, etc. but almost no information about specific proper nouns. Cyc by design should go out to appropriate databases and websites and web services to obtain whatever specific information it needs as it reasons.

Over the last 30 years, this has been expanded to encompass various domain-dependent rules and assertions in various fields, such as cell biology, oil well asset operations, and network security. These are represented in symbolic logic—mostly first-order predicate calculus plus some second- and higher-order extensions to allow Cyc to have assertions that talk about other assertions, talk about predicates (relations), talk about the problem solving process to date, and so on. The Cyc inference engine can do general deduction (via Resolution) (Melis and Siekmann 1999) but, over the years, over a thousand special-purpose reasoners (many with their own special-purpose representations) have been added in as independent agents. Together, this community of agents tackles each problem and each sub-problem; the general theorem prover always *could* apply, but 99 % of the time one of the other reasoners can jump in and solve or simplify the problem instead, in vastly less time. Cyc also has a natural language understanding and generation subsystem (Lenat et al. 2010), the latter of which BELLA uses to express its inferences and questions in English, and recursively to produce explanations of its multi-step lines of reasoning.

For the purposes of BELLA (see Figs. 2 and 3), Cyc has been extended with (1) the domain concepts and skills required to solve the domain math problems,[2] (2) the ability to declaratively model the human student's relevant knowledge, (3) the ability to declaratively model Elle's knowledge, and (4) pedagogical and psychological rules of thumb for controlling Elle's interactions with the human student, rules inferring updates to the human student model (based on that user's choices, actions, and even their *in*action), and rules for updating Elle's model based on changes to the user model.

Cyc reasons by producing all the pro- and con- arguments it can for a proposition, and then comparing and weighing them to decide which ones trump which other ones, etc., and eventually whether to believe the proposition or not. This makes exceptions straightforward to handle (Rover is a dog; dogs have four legs; but Rover happens to have three legs), along with defeasible defaults (birds fly; penguins don't; but penguins are birds).

Cyc's KB is divided into microtheories each of which has a set of contextual assumptions and some body of content; the content is internally consistent, and the assumptions help prescribe the *contexts* in which that content is asserted to hold true. For example, the assertion "Bill Clinton is President" was true in the U.S. for certain years, never true in Uganda, known by most people while it was true and since then, not known by anyone beforehand, etc. This *context mechanism* allows for the existence of different, contradictory mental models, sets of beliefs and opinions, different strategies and tactics for tackling problems, etc.—all of which are used in BELLA.

Math word problems—and the in-game situations they describe—are represented in Cyc as declarative expressions with open variables in predicate calculus using the Cyc language (CycL). Correct mental models for Elle were developed, and the various incorrect models are generated (in some cases by hand, in some cases automatically) by extirpating portions of the KB. For a particular math problem, BELLA can block Elle from solving it by going through each Cyc argument in each solution to the problem and either (i) removing at least one assertion[3]; in effect these represent gaps which Elle needs to fill (at least one of), or (ii) adding in an incorrect assertion which overrides the correct still-present ones; in effect these represent misunderstandings and errors in Elle's mental model. The type-(i) are much easier to generate mechanically than the type-(ii) which are still generally written by us by hand. As with model-tracing ITS's (Aleven et al. 2006; Anderson et al. 1995; VanLehn et al. 2000), BELLA explicitly represents and tries to recognize common misconceptions.

BELLA often gives Elle the same *types* but *more exaggerated* deficiencies than the user (which in turn it gets from the user model). Elle then manifests this exaggerated

[2] In general, Cyc already knew this; for example, how to efficiently and correctly solve "$\mathbf{3x}\text{–}\mathbf{2} = \mathbf{1}\frac{4}{9} + \mathbf{2x}$". But for BELLA, we had to go back and give Cyc explicit scripts by which it could manipulate and solve such equations step by step by step, based on laws and rules (and do that step by step process *incorrectly* when given a flawed set of laws and rules assigned to the then-current Elle mental model).

[3] Due to the robustness of Cyc's understanding of the material, it is sometimes necessary to remove a piece of domain knowledge, re-ask the same query, see that Cyc still found a way to solve the problem (correctly) in a different way, remove one of the supports of that chain of reasoning, and repeat this process until finally Cyc fails to solve the problem.

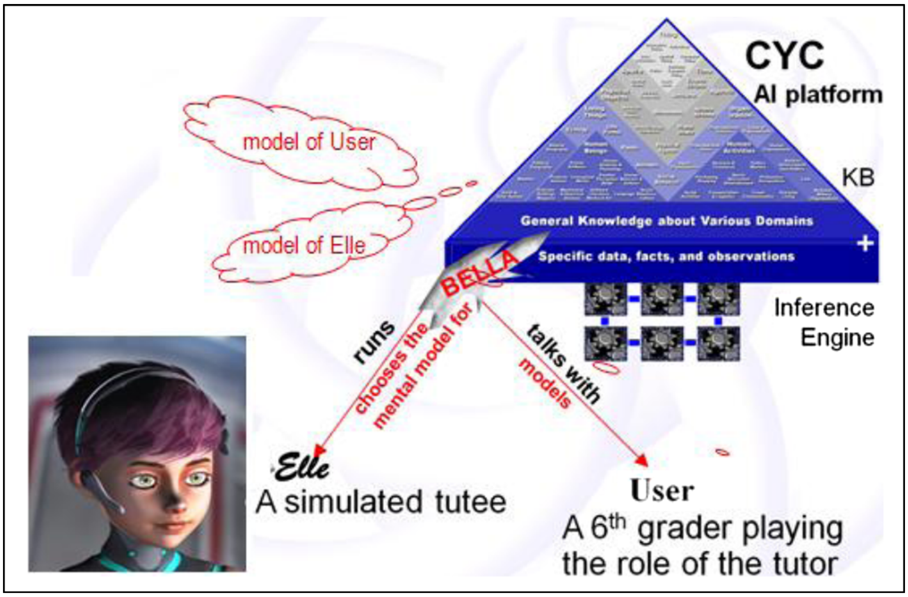
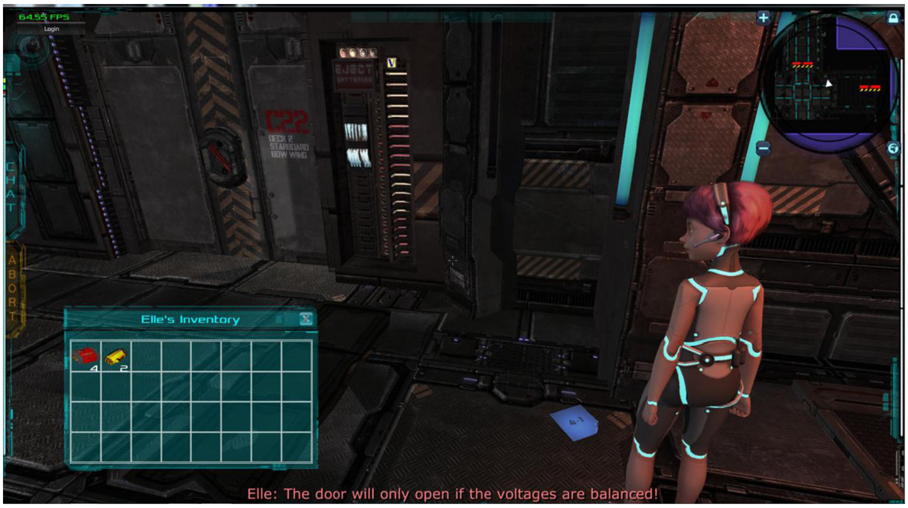

CYC
AI platform
KB
General Knowledge about Various Domains
Specific data, facts, and observations
+
Inference Engine
model of User
model of Elle
BELLA
runs
chooses the mental model for
talks with
models
Elle
A simulated tutee
User
A 6th grader playing the role of the tutor

Fig. 2 Percentage-wise, BELLA is a small expansion of Cyc's KB (about 1/10,000th) and inference engine (about 1/1000th). BELLA knows 6th grade math, but controls an avatar, Elle, who is very shaky on those concepts and skills. User is a 6th grader playing the role of tutor to Elle, but Elle *never directly learns* from User. Instead, BELLA observes what User says and does, thereby *models* them, and uses that to select *useful* flaws and gaps for Elle to have in her domain model



Elle: The door will only open if the voltages are balanced!

Fig. 3 Our "3D sandbox role-playing game"-style UI. Note the mini-map (*upper right*) showing an overhead view of nearby corridors and rooms, clues (such as the blue post-it note on the ground which contains the expression "4x-3" scribbled on it), obstacles (such as the large electromagnetically locked door), inventory (currently holding some batteries of two different voltages), and a place to ***use*** the solution to the math problem to overcome the obstacle (in this case, placing the correct linear combination of batteries into the slots just to the right side of the door, so as to sum to the voltage needed which will open this door.) Trial and error works, here, of course, but only after a long period of wasted effort (which Elle complains about.)

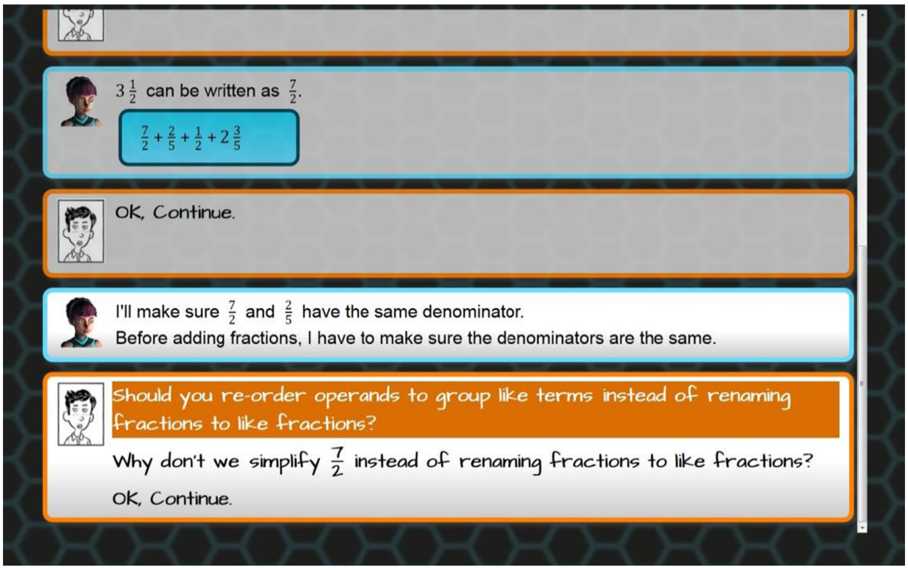
errorful behavior, thereby providing opportunities for the user to coach Elle on concepts and skills just at the leading edge of their own knowledge. As the tutoring session continues, the user's actions cause BELLA to update its knowledge model of him or her; and that in turn causes BELLA to adjust Elle's knowledge model, so that Elle's understanding (and lack thereof) keeps tracking just behind the user's.

If the user notices a mistake made by Elle, especially right away, then BELLA is more likely to assess the user as now better understanding that concept/skill; that usually is enough for BELLA to modify Elle's model so that Elle will be *less likely* to make that same type of mistake. Eventually, BELLA gives Elle a model that simply *doesn't* make that type of mistake. As the user collaborates with Elle, Elle's performance will incrementally increase (as evidenced by problem solving ability); but, the important effect is not that the software agent Elle becomes increasingly competent, but that the user*'s* understanding of that material increases as the tutoring of Elle proceeds.

If the user does not notice that Elle made a mistake, or the user seems shaky about what mistake exactly was made, then BELLA is more likely to keep Elle with that erroneous mt (microtheory; specifically, the Cyc microtheory representing her mental model), or even give her a worse mt so she will manifest that same type of error more often and more obviously (e.g., where the problem is much simpler, or the outcome is absurd). If the user still does not notice that Elle made a mistake, Elle herself might evidence some self-reflection and self-questioning.

The **student model** comprises about 150 tuples, each of which corresponds to either a Common Core concept (such as the associativity or commutativity of addition), or a Common Core skill (such as how to multiply two fractions), or a specific type of "bug" or "missing heuristic" which students may have. An example of a *bug* is the student believing they should work through all the multiplications in an arithmetic expression left to right before doing any divisions, which leads to the wrong answer in problems such as this one: **1½÷3×2**. An example of a *missing heuristic* is failing to notice that reordering addends can drastically simplify solving problems such as illustrated in Fig. 4.

Each of these 150 model elements is summarized as a number from 0 (meaning that the student definitely does not understand this concept, definitely cannot do this skill, never avoids this bug) to 100. We referred to each element as a "tuple," above, not just a scalar percentage probability, because BELLA stores the entire history of every model element: the 0–100 value at which it started (for that student), and every incident—every action or inaction by the user—which caused that number to go up or down. The teacher has access to that full history through a UI. For example, the user (here anonymized to "Barry Chen") made a good choice in the situation depicted in Fig. 4a (selecting an excellent piece of advice out of the three possible feedback lines of dialogue to respond back to Elle); that action caused BELLA to increase its certainty of belief that Barry does deeply understand "reordering operands to opportunistically group like terms" (in this case, terms with the same denominators.) BELLA then chose to modify *Elle's* mental model to permit Elle to understand and recall and employ that reordering-addends heuristic, and Elle therefore appears (Fig. 4b) to benefit from the User's suggestion, to take that suggestion and judiciously reorder the addends. That particular user model element had been initially set to 0.84 for Barry; he then went through the event depicted in Fig. 4a and b, where he made the choice which caused BELLA to increase that model element to 0.89, which is where it happens to currently be at the time that screenshot 4c is taken. If the teacher clicks on the indicated 0.89 cell, a



$3\frac{1}{2}$ can be written as $\frac{7}{2}$.

$\frac{7}{2}+\frac{2}{5}+\frac{1}{2}+2\frac{3}{5}$

OK, Continue.

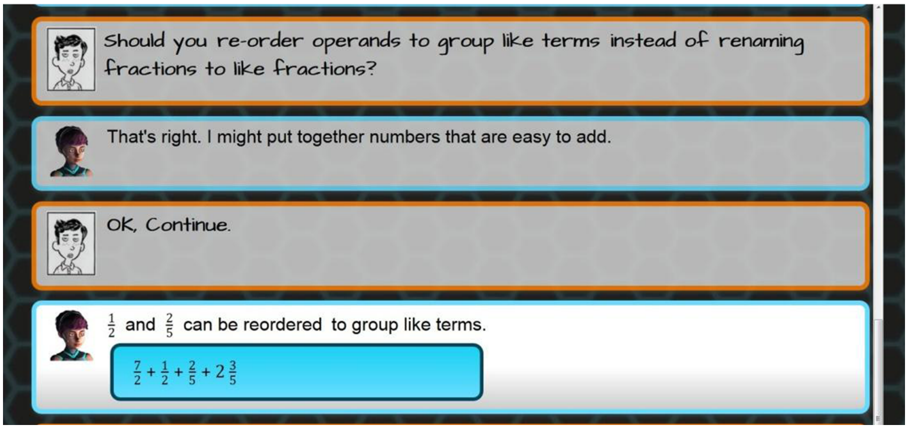
I'll make sure $\frac{7}{2}$ and $\frac{2}{5}$ have the same denominator.
Before adding fractions, I have to make sure the denominators are the same.

Should you re-order operands to group like terms instead of renaming fractions to like fractions?

Why don't we simplify $\frac{7}{2}$ instead of renaming fractions to like fractions?

OK, Continue.

b



Should you re-order operands to group like terms instead of renaming fractions to like fractions?

That's right. I might put together numbers that are easy to add.

OK, Continue.

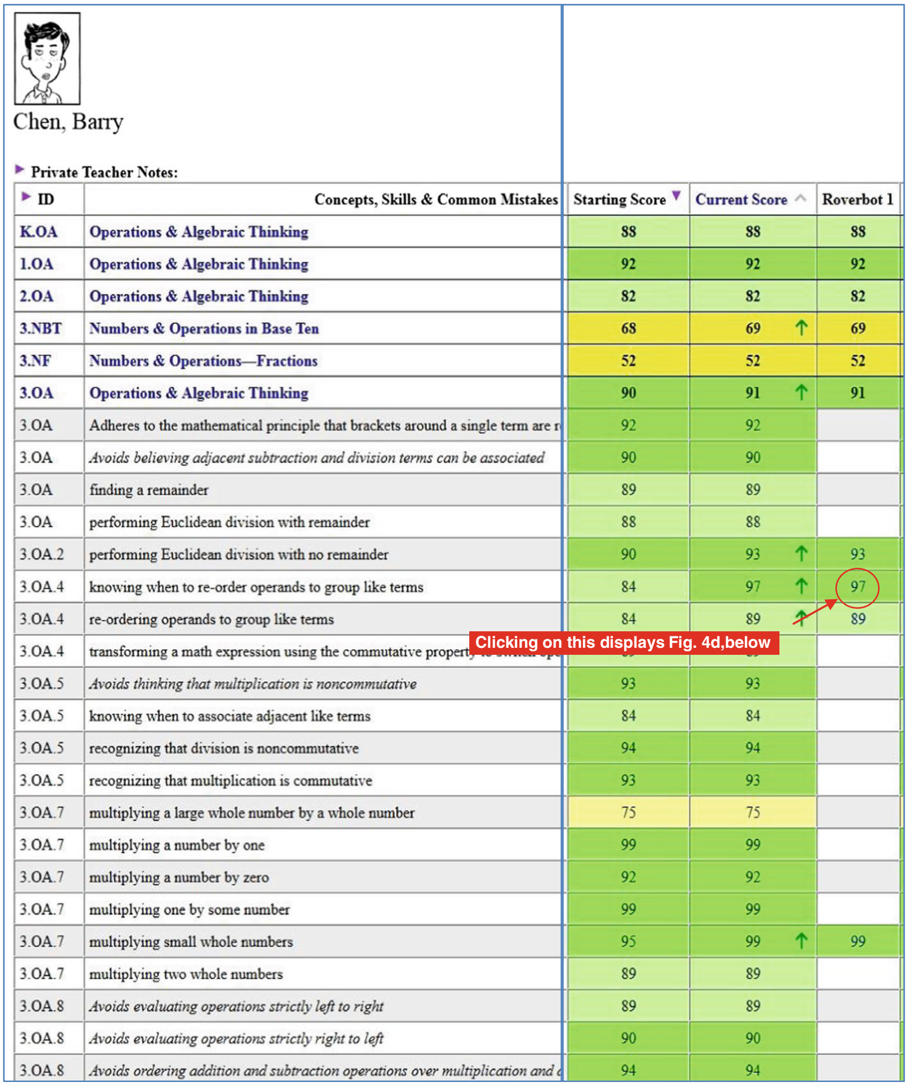
$\frac{1}{2}$ and $\frac{2}{5}$ can be reordered to group like terms.

$\frac{7}{2}+\frac{1}{2}+\frac{2}{5}+2\frac{3}{5}$

Fig. 4 **a** When a math problem begins, the Math Panel window opens. Elle performs a step, and the user then chooses from a set of feedback options (*orange highlight* when hovered over) for the player to give to Elle. In this case, Elle was about to doggedly add those four numbers in order, left to right, without realizing that reordering them makes the whole problem much simpler. User is about to select that piece of advice; the second possible feedback option is a red herring; the third ("OK. Continue.") represents *inaction* at this point by the user. **b** Elle ***appears*** to remember that addends can be reordered in this way ("That's right, I might put together numbers that are easy to add"), and takes the User's suggestion. Behind the scenes, what's really happening is that BELLA decides that the User has shown they understands this concept, hence so should Elle. **c** 13 of the 150 model elements for Barry Chen (*rows*), and the history of all changes to those elements (*columns*). The events of **a-b** occur while Elle is solving Roverbot problem 1. **d** The teacher drills down to see when/why this change was made (by clicking on the 0.97 in the *red circle* in **c**), and sees this historic record of the events (**a-b**) that caused BELLA to make that change in that model element of that user, Barry, raising it by + 0.13, from 0.84 to 0.97

a tab opens up (Fig. 4d), displaying the exact historical in-game situation—and user actions/choices—which caused BELLA to revise that model element.

c



Chen, Barry

Private Teacher Notes:

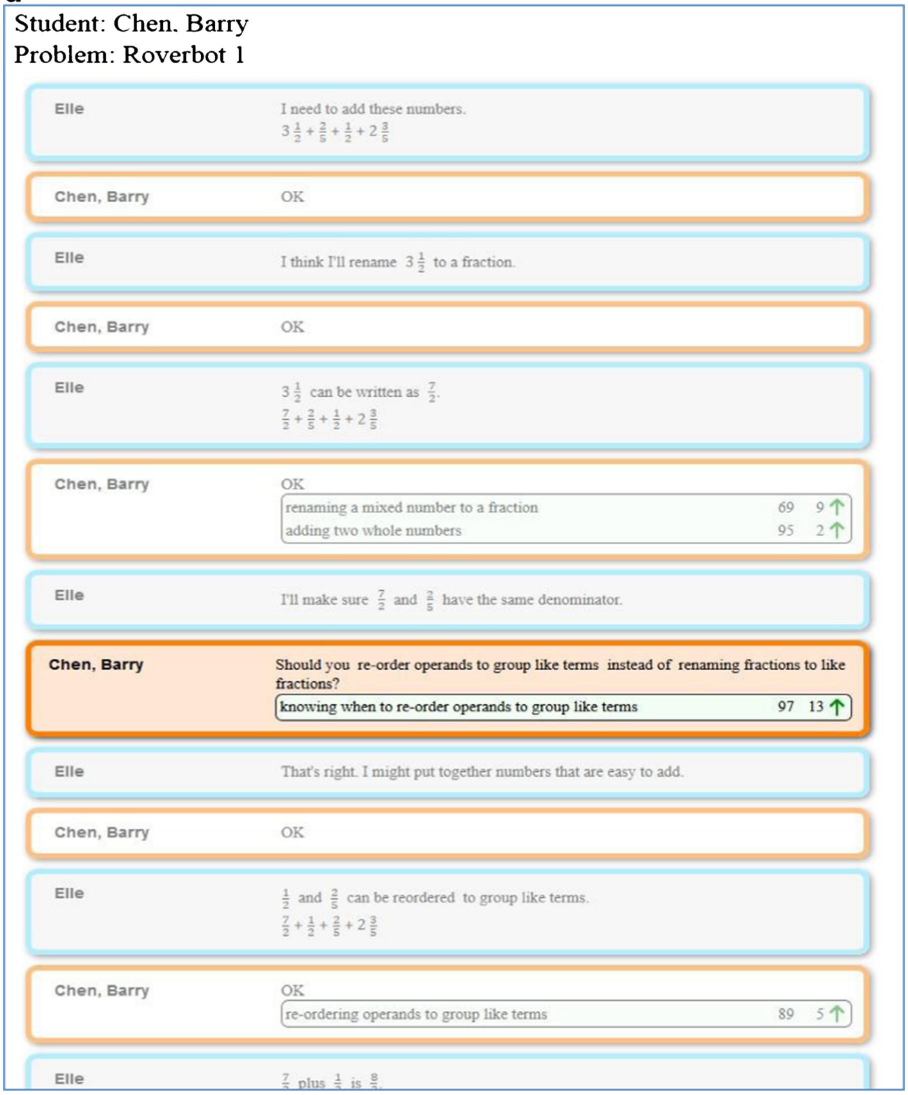
ID	Concepts, Skills & Common Mistakes	Starting Score	Current Score	Roverbot 1
K.OA	**Operations & Algebraic Thinking**	**88**	**88**	**88**
1.OA	**Operations & Algebraic Thinking**	**92**	**92**	**92**
2.OA	**Operations & Algebraic Thinking**	**82**	**82**	**82**
3.NBT	**Numbers & Operations in Base Ten**	**68**	**69** ↑	**69**
3.NF	**Numbers & Operations—Fractions**	**52**	**52**	**52**
3.OA	**Operations & Algebraic Thinking**	**90**	**91** ↑	**91**
3.OA	Adheres to the mathematical principle that brackets around a single term are r	92	92	
3.OA	*Avoids believing adjacent subtraction and division terms can be associated*	90	90	
3.OA	finding a remainder	89	89	
3.OA	performing Euclidean division with remainder	88	88	
3.OA.2	performing Euclidean division with no remainder	90	93 ↑	93
3.OA.4	knowing when to re-order operands to group like terms	84	97 ↑	97
3.OA.4	re-ordering operands to group like terms	84	89 ↑	89
3.OA.4	transforming a math expression using the commutative propert[illegible]	[illegible]	[illegible]	
3.OA.5	*Avoids thinking that multiplication is noncommutative*	93	93	
3.OA.5	knowing when to associate adjacent like terms	84	84	
3.OA.5	recognizing that division is noncommutative	94	94	
3.OA.5	recognizing that multiplication is commutative	93	93	
3.OA.7	multiplying a large whole number by a whole number	75	75	
3.OA.7	multiplying a number by one	99	99	
3.OA.7	multiplying a number by zero	92	92	
3.OA.7	multiplying one by some number	99	99	
3.OA.7	multiplying small whole numbers	95	99 ↑	99
3.OA.7	multiplying two whole numbers	89	89	
3.OA.8	*Avoids evaluating operations strictly left to right*	89	89	
3.OA.8	*Avoids evaluating operations strictly right to left*	90	90	
3.OA.8	*Avoids ordering addition and subtraction operations over multiplication and d*	94	94	

Fig. 4 (continued)

All of these specific event drill-downs (Fig. 4d) are available to the teacher; they aggregate into the student model (Fig. 4c), and all the students in that teacher's classes are further aggregated and summarized for them in the top-level UI for that teacher (Fig. 5)

One of our hypotheses is that it suffices, for the BELLA task, to model each student's (and Elle's) domain knowledge, at any point in time, as no more and no less than this set of about 150 Common Core elements and "known bugs and heuristics", and to keep around the historical context for each change that was made to each of those numbers. By the time we have fully covered the sixth grade Common Core Curriculum, we estimate that the size of the user model (and hence the Elle model) will be about 200 tuples.

d

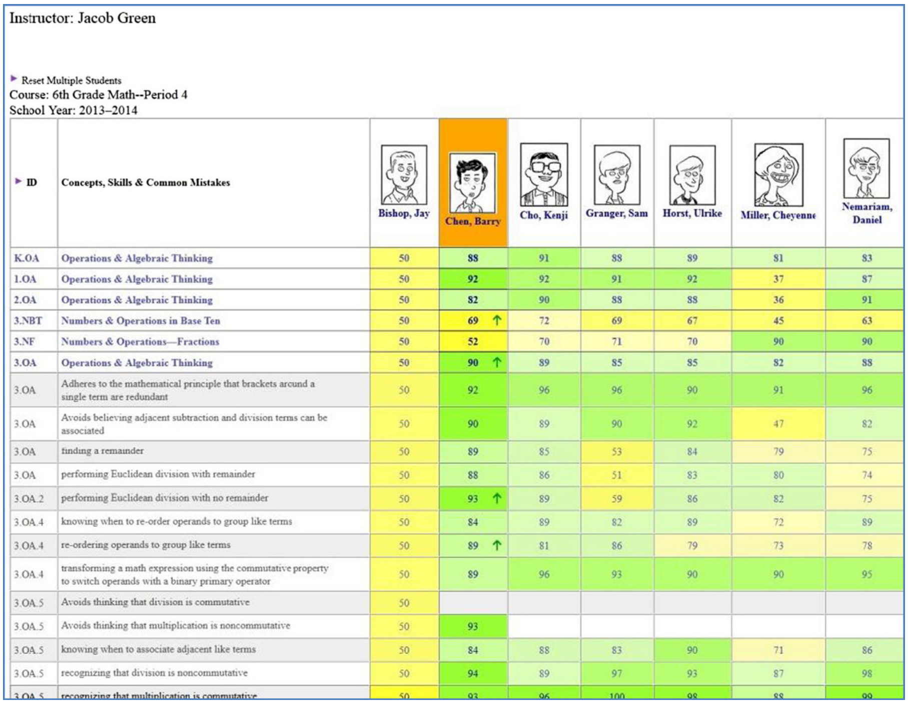


Student: Chen, Barry
Problem: Roverbot 1

Elle	I need to add these numbers. $3\frac{1}{2}+\frac{2}{5}+\frac{1}{2}+2\frac{3}{5}$
Chen, Barry	OK
Elle	I think I'll rename $3\frac{1}{2}$ to a fraction.
Chen, Barry	OK
Elle	$3\frac{1}{2}$ can be written as $\frac{7}{2}$. $\frac{7}{2}+\frac{2}{5}+\frac{1}{2}+2\frac{3}{5}$
Chen, Barry	OK renaming a mixed number to a fraction 69 9 ↑ adding two whole numbers 95 2 ↑
Elle	I'll make sure $\frac{7}{2}$ and $\frac{2}{5}$ have the same denominator.
Chen, Barry	**Should you re-order operands to group like terms instead of renaming fractions to like fractions?** **knowing when to re-order operands to group like terms 97 13 ↑**
Elle	That's right. I might put together numbers that are easy to add.
Chen, Barry	OK
Elle	$\frac{1}{2}$ and $\frac{2}{5}$ can be reordered to group like terms. $\frac{7}{2}+\frac{1}{2}+\frac{2}{5}+2\frac{3}{5}$
Chen, Barry	OK re-ordering operands to group like terms 89 5 ↑
Elle	$\frac{7}{}$ plus $\frac{1}{}$ is $\frac{8}{}$.

Fig. 4 (continued)

While we have not settled firmly on the best way to *initialize* the student model, the methods we are considering are: (a) pretesting, (b) starting with a normative 6th grade model, (c) interviewing the teacher (about this student), (d) interviewing the student, (e) starting with a very incompetent Elle (which will quickly get better, to match the student), or—most likely—some combination of these. One interesting expected (not yet tested) advantage of the "very incompetent Elle" starting model is that the user should easily catch and correct several mistakes of Elle's very early on, which may in turn may build up the user's confidence and self-esteem, and may reinforce the user's belief that Elle needs tutoring and more generally needs their help.



Instructor: Jacob Green

▸ Reset Multiple Students
Course: 6th Grade Math--Period 4
School Year: 2013–2014

▸ ID	Concepts, Skills & Common Mistakes	Bishop, Jay	Chen, Barry	Cho, Kenji	Granger, Sam	Horst, Ulrike	Miller, Cheyenne	Nemariam, Daniel
K.OA	Operations & Algebraic Thinking	50	88	91	88	89	81	83
1.OA	Operations & Algebraic Thinking	50	92	92	91	92	37	87
2.OA	Operations & Algebraic Thinking	50	82	90	88	88	36	91
3.NBT	Numbers & Operations in Base Ten	50	69 ↑	72	69	67	45	63
3.NF	Numbers & Operations—Fractions	50	52	70	71	70	90	90
3.OA	Operations & Algebraic Thinking	50	90 ↑	89	85	85	82	88
3.OA	Adheres to the mathematical principle that brackets around a single term are redundant	50	92	96	96	90	91	96
3.OA	Avoids believing adjacent subtraction and division terms can be associated	50	90	89	90	92	47	82
3.OA	finding a remainder	50	89	85	53	84	79	75
3.OA	performing Euclidean division with remainder	50	88	86	51	83	80	74
3.OA.2	performing Euclidean division with no remainder	50	93 ↑	89	59	86	82	75
3.OA.4	knowing when to re-order operands to group like terms	50	84	89	82	89	72	89
3.OA.4	re-ordering operands to group like terms	50	89 ↑	81	86	79	73	78
3.OA.4	transforming a math expression using the commutative property to switch operands with a binary primary operator	50	89	96	93	90	90	95
3.OA.5	Avoids thinking that division is commutative	50						
3.OA.5	Avoids thinking that multiplication is noncommutative	50	93					
3.OA.5	knowing when to associate adjacent like terms	50	84	88	83	90	71	86
3.OA.5	recognizing that division is noncommutative	50	94	89	97	93	87	98
3.OA.5	recognizing that multiplication is commutative	50	93	96	100	98	88	99

Fig. 5 All the students in this teacher's sixth grade math classes are summarized at this top-level interface. Clicking on Barry Chen's column, e.g., opens up the interface showing how each of these 0–100 numbers got to where it is (Fig. 4c). Clicking on a number *there* drills down to show the detail in Fig. 4d

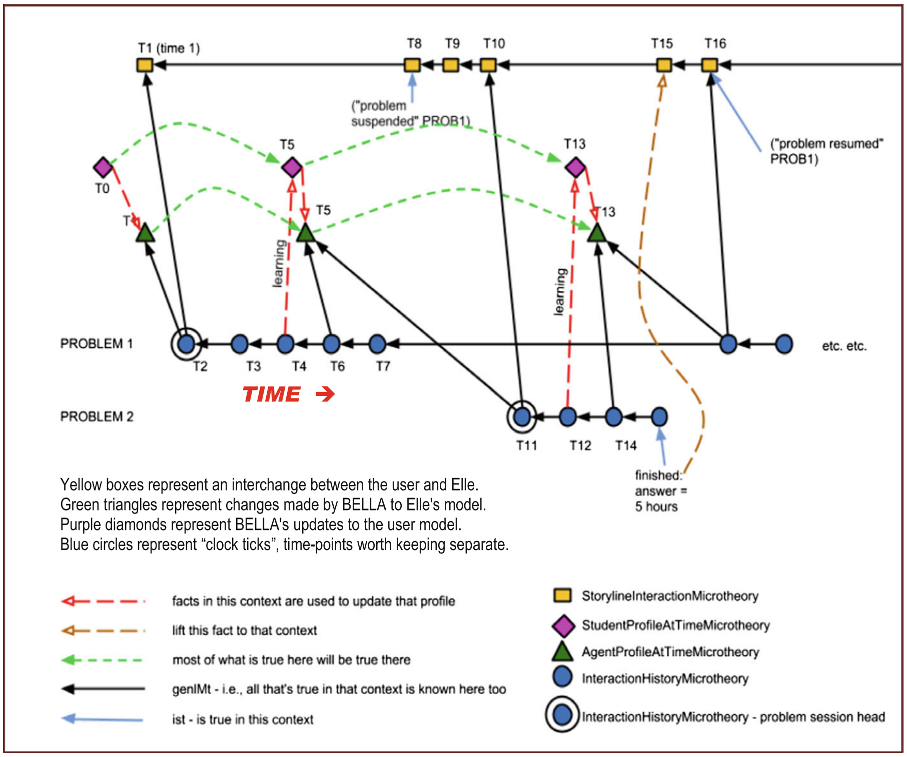
In parallel with the above steps, the user interface and the adventure game context for BELLA had to be designed. Our original UI conception (upon which the Clued-In game (Fig. 1) was based) turned out to be far too complicated, offering hundreds of actions a user could take at any time. Our new streamlined UI (Fig. 3) is less daunting and more familiar to users who play video games. Elle moves through a virtual environment facing problems and challenges. When solving problems, Elle pauses after each step, allowing input from the user (Fig. 4a and b). The most frequent input is just "Okay. Continue on" but the most telling events occur when the user chooses one of the other optional responses instead. At each step, BELLA selects the most productive three or four choices to present to the user, based on the current situation (current state of the problem, current student model, and Elle's current model). BELLA decides whether to change the Elle model, based on what the user just said (or didn't say), and Elle responds—sometimes disagreeing with the user or appearing (due to gaps and errors in Elle's model) to not even understand the user's suggestion.

Looking Under the Hood

Microtheories in BELLA As Elle and the user work a problem together, there might be dozens of these "memorable interactions." Every time the user or Elle says or does something, that event spawns a new reified context which is a slight extension of the previous context. In the new context, time will have advanced one "tick", and a few things might be true that used to be false or vice versa, such as the state of the problem, or Elle's model, or the user model. This results in a rich tapestry of dozens of new microtheories generated in a small amount of time (see Fig. 6), and one of the challenges to Cyc's inference engine in early 2013 was how to keep up with this proliferation in real time. The short answer is that this was accomplished by pre-creating and caching a large resource pool of such mts.

As our example (in the next section) illustrates, this structure enables Elle to plan its future actions; it enables Elle and the user to refer to earlier states in their dialogue and access the history of what each of them said and did; and it allows BELLA to reason partially about what the user could and should have known—and precisely, totally, about what Elle *did* know—in each past context. Cyc already had the capability to represent these types of knowledge and do these types of reasoning; one of the benefits of constructing BELLA as an extension to Cyc was its being able to utilize this machinery.

BELLA maintains and uses several types of mts; a few of these mt-types are depicted in Fig. 6. Each green triangle mt states (or inherits) assertions representing what Elle believes about the current state of the world in general, the relevant math concepts and skills in particular, and the current state and history of it (Elle) working on


T1 (time 1)
T8
T9
T10
T15
T16
("problem suspended" PROB1)
("problem resumed" PROB1)
T0
T5
T13
T
T5
13
learning
learning
PROBLEM 1
T2
T3
T4
T6
T7
etc. etc.
TIME →
PROBLEM 2
T11
T12
T14
finished: answer = 5 hours
facts in this context are used to update that profile
lift this fact to that context
most of what is true here will be true there
genlMt - i.e., all that's true in that context is known here too
ist - is true in this context
StorylineInteractionMicrotheory
StudentProfileAtTimeMicrotheory
AgentProfileAtTimeMicrotheory
InteractionHistoryMicrotheory
InteractionHistoryMicrotheory - problem session head

Yellow boxes represent an interchange between the user and Elle.
Green triangles represent changes made by BELLA to Elle's model.
Purple diamonds represent BELLA's updates to the user model.
Blue circles represent "clock ticks", time-points worth keeping separate.

Fig. 6 As Elle and the user interact, a growing graph of interaction contexts is built. They snapshot the user model at that moment, the Elle model, the state of the problem (and subproblem, etc.) being worked on, etc. This provides a way for Elle or the user to refer to past states of their dialogue, to back up a step or two, and grist for BELLA to induce short- and long-term conclusions about the user

this problem. Each purple diamond mt contains a temporal snapshot of the user model—i.e., the things which the user believes about the world (more accurately, what BELLA believes the user to believe.) Blue circle mts represent time points of interest where user actions, planning steps, and Elle's chat statements and Math panel actions occurred.

There are also several types of *timeless* mts (which therefore do not appear in Fig. 6): *Process* mts each represent a script, a particular algorithm—which might be correct or erroneous—for how to carry out a procedure such as adding mixed fractions. *Math belief* mts involve facts that might be required when reasoning through a problem, such as order of operations (including potential erroneous "facts"). *Quantitative relationship* mts represent formulas, such as the formula for the area of a circle, or the formula relating time, speed, and distance (again, including possible erroneous formulas). *Problem interpretation* mts represent targets for Elle to understand game-world situations (or word problems), i.e., how to set up a mathematical problem or equation given what the game-world challenge or opportunity is. *Tactician* mts represent alternative plans of attack (possibly errorful) on a type of problem, and *strategist* mts represent rules for selecting a tactician mt and deciding when to suspend the current one and try a different plan of attack on the problem.

With regard to computational steps, e.g., adding two fractions, or finding the area of a circle, the steps can be carried out in multiple ways, but many students have learnt only one way. Cyc has no trouble representing alternative methods (both correct and erroneous), as separate mts (e.g., Process mts). In BELLA, some of the erroneous mts were created by hand, and added to the Cyc KB; gradually, a larger and larger fraction have been produced automatically by introducing systematic errors into a correct mt, much as Brown and VanLehn (1980) did with DEBUGGY subtraction flowcharts.[4]

Elle's assigned mts are examined during her reasoning by the use of *articulation axioms,* also called *lifting rules* (Sowa 1995). These are if/then rules that specify which mt a premise is to be satisfied in, or which mt a conclusion is to be recorded as holding true in. Process mts are not generally visible to the reasoning mts described above, but the other mts are, reducing the amount of lifting that BELLA needs to do.

Natural Language Understanding and Generation Before starting the example, we want to remark on the method of interaction in what appears to be readable English, that occurs back and forth between the user and Elle (in reality, in BELLA, not in the fictional universe!):

- NLG: Elle comments many of her math problem steps with chatty English commentary (such as, in Fig. 4a, "Before adding fractions, I have to make sure the denominators are the same"). These are produced by BELLA using Cyc's natural language generation (NLG) subsystem (Lenat et al. 2010) built around a semantic construction grammar (SCG) comprising about 200,000 rules and assertions. These recursive templates do a good job of converting Cyc's internal predicate calculus representation into understandable English. In cases of multi-step explanations, Cyc has a moderate ability to combine the sentences generated and introduce pronouns

[4] Though that is not a focus of our research, we note that, as with their research, we have found some overlap between automatically generated errorful mts and the common math errors that sixth graders actually make.

etc. so that they don't sound quite so artificial. Fortuitously, most users just attribute any residual stiltedness in the generated sentences to the fact that Elle is a robot.

- NLU: Despite 60 years of progress, *understanding* unrestricted natural language (NLU) is far from a solved problem. In BELLA, we sidestep this by providing the user with gestures (left-click on places or things in the game world; see Fig. 3) and—especially while in the process of solving a particular math problem—a menu of responses to choose from (see Fig. 4a and b). At each step (Fig. 4a-b), BELLA formulates five or ten possible choices for what the user might want to say to Elle. It does this by running a set of alternative-generator rules which apply broadly, augmented here and there with hand-formulated alternatives for pivotal decision-points in the game. For example, one banal alternative-generator rule says "Whenever a mixed number is involved, suggest converting it to a fraction". Instead of overwhelming the user with long menus, BELLA invokes meta-rules to decide which three or four of those alternatives are best: most useful at further revealing details of the user's mental model, or most useful at helping the user see how to correct something that Elle is doing wrong. So the user usually sees just two or three revealing alternatives to select among.

Example Problem Scenario

This section provides more details about BELLA, presented in the context of an extended example, which in turn is motivated by a brief précis of the adventure game which we have built around BELLA, and which from the user's point of view *is* "the BELLA application". For even more detail about how/why the math problems in this example (setting up and solving a few univariate linear equations) arise naturally during the gameplay, please see Appendix 1; for even more examples of the sorts of math problems in BELLA and how they arise naturally, please see Appendix 2.

The game backstory situates the user as a 12 year old living in 2265.[5] Elle is a robot companion permanently linked to the user mentally. The starship carrying Elle and tens of thousands of other inactive companions is attacked and crashes onto a planet's surface. All the equipment to activate them is damaged, but the user had just (coincidentally, and against the rules) woken Elle up just before the crash. So Elle finds herself stranded, in a dangerous situation, with only a mental link to the user as outside communication. Elle must traverse multiple challenges at ever-increasing "scale": exploring and exploiting the crashed ship, the nearby area, adversaries, buried alien ruins, etc.—eventually triumphing by repairing the ship and awakening all the other robots.

In the specific example challenge which we elaborate here, Elle needs to get past a series of electro-magnetically-operated doors which are currently not working due to

[5] The students did not like our initial choice of storyline (shopping, cooking, construction; they wanted futuristic sci-fi, robots, and aliens), "lame" character names, confusing cluttered interface design, and unmotivated word problems. The design and examples presented here represent our "Year 2" versions of all these.

main ship's power being out. The example illustrates (1) how Elle's current mental model impacts her next action, (2) how the user input leads to revision of BELLA's model of the user, (3) how the user model affects which menu choices the user is offered. Appendix 2 presents seven more examples of game-emergent math problems.

The Example The story so far[6]: Elle has learned that a 227 V door could be opened by connecting (in series) 11 20-V batteries plus three small unmarked batteries. The next door she encounters is labeled "228 V". Ten 20 V batteries are fused to the wall next to the door, already permanently connected to the door. There is a pile of five[7] of the now-familiar but *unmarked* small batteries here. If the user has Elle connect these, the door is still underpowered and doesn't budge. Elle says: "I need more small batteries. But, how many? I guess it depends on how much voltage each one supplies." If the user doesn't head back to the previous (227v) door on their own, Elle says "Let's go back to the last room to check it out." The user moves Elle back up the corridor to the room with the 227v door. Elle says, "Let's figure this out. To get to 227 V, I used 11 20 V batteries here and three small batteries." At that point, the Math Panel (similar to Fig. 4a and b) pops up; after the problem is solved (or the user decides to give up on it and sidestep it, or guesses it by trial and error), that panel disappears.

For most users, Elle sets up the problem correctly, namely as **(11 * 20 V)+3x= 227 V,** and works through it step by step. The user sees what Elle writes on the Math Panel, along with her narration of what she's doing (see "NLG:", above). At each step, the user can click on the "Okay" option, and Elle just moves on to the next step. The alternatives are things that the user could say to Elle at that moment, ostensibly to tutor it to better understand what it's doing wrong. Some of those tutoring options are good, some are bad, but all of them are revealing in the sense that each and every choice enables BELLA to extend, refine, and update its model of this user at this time.

For the sake of this discussion, let us conceptually separate

- Game: the adventure game (running Unity 3D)
- UI: the user interface (running Firefox) displaying the Math Panel and Teacher Panel
- Cyc: the knowledge base and reasoning system (running Java) which declaratively represents and modifies the user models and the Elle models and the underlying math.
- User: the current sixth grader logged into the system; let's continue to call him Barry.

So Game sends Cyc a message that this particular math problem is being started. Elle must now decide—meaning that Cyc must now decide—on Elle's strategy and

[6] Appendix 1 goes through the first part of this scenario in much greater detail.
[7] These variables—227, 228, 20, 10, 5, etc.—are chosen by BELLA at the time the user enters this series of corridors, based on elements of the user model such as their ability to multiply numbers of various magnitudes, and problem specific constraints such as the door voltages not being a multiple of the large battery voltage.

tactics for approaching this type of problem. Of course Cyc could solve this problem right away, correctly, but that's beside the point! *Elle's* mental model is now what is being "run", not Cyc's, and *Elle's* current mental model has gaps, errors, shakiness, etc. In this case, suppose that Cyc's current model of Barry indicates that he is moderately skilled at setting up simple algebraic expressions. Based largely on that, Cyc has allowed Elle's current model to correctly recognize this as the *type* of problem in Cyc's ontology called "a moderately good student solving a linear equation in one unknown". Elle believes (correctly, it turns out) that she can solve this type of problem in three phases: (1) simplify the equation as much as possible, then (2) isolate the terms containing the variable $\boldsymbol{x}$ from the non-$\boldsymbol{x}$ terms, and then (3) multiply or divide each side of the equation by the right scalar to turn the variable-term side into just plain $\boldsymbol{x}$**,** at which point the equation will read "$\boldsymbol{x}$ = <number>" and she believes she will be done.[8]

Cyc sends UI the information to fill in the first "step" on the Math Panel, stating the problem and explaining in English why this problem is being worked on, why it was set up the way it was. In this case, the equation to display on the Math panel is **(11 * 20 V)+3x=227 V**, and the string to display before it is "This is the equation we need to solve." Cyc's NLG produces this string to display after the equation: "11 20 V batteries plus three small batteries total to 227 V."

Cyc now has the goal of communicating this plan (currently represented as a set of predicate calculus formulae) to the user. Cyc's NLG subsystem translates the predicate calculus for the first phase of the plan into English as: "I need to simplify and get the x term by itself over on the left side."[9] Cyc sends this to the UI, and the UI displays it on the Math Panel.

Then, step by step, Cyc sends the UI specific information about what MathML (www.w3.org/Math/) it should display and what NLG comments (if any) it should display. For each step—each lozenge of the sort depicted in Fig. 4a and b—Cyc creates and names a brand new interaction context. Let's say the next interaction context (which is a microtheory or mt in Fig. 6) gets the name "Step37".

Cyc runs the Elle model, executing the next step (in this case the *first* step) in the declarative script which this current Elle will follow, to solve the problem. That script says to discharge expressions in parentheses first if possible, so Elle takes the step of replacing (11 * 20 V) by 220 V. This is indeed one step toward simplifying the equation. Cyc sends to the UI web service the chat message "Eleven times twenty is 220" with an associated "STEP37" tag for alignment purposes, and a Math panel message containing the MathML for "**220 V+3x=227 V**" also with the tag "Step37". The user sees that equation and that comment and the list of possible reactions to say back to Elle, one of which is always just "Okay. Continue on" (for consistency, this is always the last option listed). To decide on the set of most salient options,

[8] If it ends up as "<number> = x", this slightly rigid model of Elle will believe that the problem is not yet solved, and take a whole extra step to change it into the "$\boldsymbol{x}$ = <number>" form.

[9] For efficiency reasons, this part of Cyc's work gets performed at the very start of ZONE3, just before the first "electromagnetic door"—i.e., back at the same time that various parameters (227, 220, 20, 228, 11, 5, etc.) were chosen.

BELLA uses the current user model (and to a lesser extent prior models of this user), prioritizes them, picks the top-rated few, and sends a message to the UI:
<Choice List of options STEP37

- **How did you calculate that 11×20 V is 220 V?**[10]
- **Why do that first?**[11]
- **Can we multiply volts times non-volts?>**[12]

After those three starred options, the game/UI automatically displays the "**Okay**" option as well.

Not all users will see the "Can we multiply volts times non-volts" option. Something that Barry has done in the past, or failed to do in the past, impacted elements of his model, and that in turn caused this option to get a higher rating than others which different users might see instead (e.g., other users might instead see this option: "Shouldn't it be '3x *volts*', there, not just '3*x*'?)

Suppose the user clicks Okay at interaction Step37. Then BELLA continues to run the Elle model, which selects the next appropriate step for Elle to do in solving this linear equation. In this case, the first *phase* was repeatedly simplifying until there's nothing more to do, and in fact after this one simplification step there *is* no more to do, so the second phase of the script that Elle is following starts: isolating "like terms" (getting all the ones with variables on one side, all the ones without variables on the other side). This it *can* perform, on the current equation, and Cyc chooses to subtract 220 V from both sides. Cyc records its decision, and composes a message to the UI (in Elle's name of course) with the MathML for "**220 V - 220 V+3x=227 V - 220 V**" for the Math panel, and a synchronized NLG message "I can subtract 220 from each side." A slightly different Elle model might have had her explicitly say "220 V" instead of just "220"; an even more different Elle model might have had her subtract a different quantity, or divide every term in the equation by 3, and so on.

Cyc's NLG could translate, into English, any or all of the symbolic logic derivation of its previous conclusion—i.e., its step by step reasoning "argument" that led it to decide to subtract 220 V from both sides. This quite long and detailed logic proof gets pared down to the one or two crucial steps in the argument, which are then translated into English (by the NLG rules), which in turn gets packed into the message format and transmitted to the UI. So, onscreen, Elle adds the comment "If I subtract the same number from each side of the equation the two sides stay equal, so it's okay for me to subtract 220 from each side. and 220–220 will simplify in the next step to zero, which I think is making progress." Let's say the user clicks "Okay" now.

[10] If chosen, Elle's model will respond "That's just multiplication." And BELLA will probably downgrade the element of the user model involving doing certain ranges of multiplications "in one's head"

[11] If chosen, Elle's model will respond "Take care of parentheses first." And BELLA may adjust the certainty of the user model element involving order of precedence of operations, particularly parenthetical expressions. If the current model value is very low, then move it upwards toward 0.50, and if it's very high then move it downwards toward 0.50

[12] If chosen, this makes an important addition to the user model, namely that this user at this moment has confused the inability to add/subtract unlike unit of measure quantities with the inability to multiply/divide unlike unit of measure quantities (the latter of which of course is fine); and a prediction that the user may get the unit of measure wrong if multiplying two quantities which have the same unit of measure. Instead of overwriting what the previous model said, a new user model is created, marked as a modified version of the extant user model, and the appropriate meta-level information is recorded about when and why this was created

The math panel advances, and a re-write of the equation appears "**0 V+3x=227 V - 220 V**." Cyc also tells the UI to include in that step's lozenge the comment "220 – 220=0".

The next two steps—in an order determined by Elle's model—remove the "0 V" addend entirely, from the left side, and simplify the right hand side to "7 V".

So at this point—call this interaction context"Step41"—the Math panel is showing the current state of the equation as: "**3x=7 V**. "This is a pivotal moment: what happens next depends in a big way on the current Elle model. A good thing for Elle to do is to divide both sides by 3. A wasteful thing to do is to divide both sides by 7. Adding or subtracting something from both sides would also be wasteful.

Let's Suppose that Elle's Current Model has a Serious Bug in it, Namely it Thinks that 3x – 3=x This is a surprisingly common bug amongst human sixth graders, by the way. The same bug causes Elle to incorrectly conclude that "7 V—3" will evaluate to 4 V. At this point in time, therefore, perfectly logically correct reasoning—given that bug in its mental model—leads Elle to believe she can make progress by subtracting 3 from each side of "3x=7 V"; i.e., Elle thinks the left hand side will turn into "x" and the right hand side will be easily evaluated and turn into "4 V".

The Math panel now shows "3x - 3=7 V – 3" Cyc NLG has told the UI to include the comment "Okay, now I'll just subtract 3 from each side." Suppose this interaction mt gets the identifier "Step42".

If user hits "Okay"—fails to correct Elle at this crucial moment—then (Cyc's simulation of) Elle blithely continues on and says in the UI "Good, we're done!" and then "Each small battery is 4 V."

The user can still recover by selecting an option to have Elle check her work, in which case she realizes she's done *something* wrong: "Let me double-check. If x=4 V, does 3x=7 V?"

"No way! Three times four is twelve, so I must have made a mistake. Let's back up a little bit." At this point, Elle decides to back up to the Step42 interaction context. There can sometimes be hysteresis learning effects, but in this case the "state" is almost exactly the same as though the user had just said something other than "Okay" back at the step where Elle wrote: "3x - 3=7 V - 3".

So suppose at the Step42 mt (step, interaction context, lozenge,…) above, the user had chosen, instead of "Okay", one of the other set of possible responses. Those possible responses are generated by Cyc by running option-generator rules, followed by running meta-rules which prioritize them, namely evaluate (i) the pedagogical value of each option choice to the user at this moment in time (based on the latest user model) and (ii) the information-gathering potential for BELLA to learn more about the user's current mental model. Cyc picks the few highest-rated choices, composes the proper message, and eventually this is presented to the user as the non-Okay choices of response in the Math panel for that step. Almost anything the user selects here will be revealing, and hence will result in updates to the user model (see Fig. 7).

Each of the four choices in Fig. 7 illustrates some interesting aspect of the BELLA system, so let's go through them one by one.

Case 1: A very subtle hint. Let's say the selected the first option, [id 100], "**Is 3x the same as 3+x?**" This is a moderately good suggestion for the user to make—i.e., it would be a good suggestion if the user were tutoring an actual human

```
<Choice List of options:
   * Is 3x the same as 3 + x?   [id 100]
   * What would happen if you subtracted 3 from each side? [id 101]
   * Should we just divide both sides by 7 instead? [id 102]
   * Should we just divide both sides by 3 instead? [id 103] >
```

Fig. 7 Cyc generates three response options besides "Okay. Please Continue." The formal predicate calculus representations get ids# 100, 101, 102, 103, and the Cyc NLG converts those into the four strings shown here, which get sent to the UI to display to the user

being. The user wants Elle to realize that *if* the left hand side of the "3x=7" equation had been "3+x" instead of "3x", *then* subtracting an integer from both sides could indeed turn the left hand side into just "x". But the left hand side is not "3+x", it's "3x".

Case 2. A moderately subtle hint. Let's say the user selected option [id101]—"**What would happen if you subtracted 3 from each side?**"—instead of [id100]. This is another good option, to get Elle to realize that subtracting 3 does not turn 3x into x. Since this involves triply-nested modal operators, let's restate this in slightly more detail: Cyc believes (logically abduces) that the User believes that Elle would believe all of the following: that subtracting 3 from both sides of "3x=7" is allowable, that the left side would then simplify to *zero* and the right side (7 -4) would simplify to 3, that the resulting equation would be "0=4", and that Elle would immediately realize that 0=4 is absurdly wrong. Representing this triply-nested modal assertion (what Cyc believes that the User believes that Elle believes), and reasoning efficiently with it, is a good example of why the power of Cyc's expressive representation language (higher order logic) and community-of-1000-specialized-reasoners inference engine are needed, in BELLA.

Depending on Elle's current mental model, Cyc might or might not have Elle carry out this line of reasoning and reach an "Aha!" moment wherein Cyc allows Elle's model to shift to correct the above bug. More specifically, Elle learns nothing directly from the user choosing response [id100] or [id101]. But Cyc learns something important, namely that the user would not have made this mistake, in this problem, and understands why. Suppose that the current user model says that Barry is somewhat shaky on these concepts, on solving univariate linear equations. Barry's wise choice id100 or id101 choice gives Cyc an argument for creating a modified user model, an updated one which says that even though Barry previously *was* shaky in that way, there is new evidence that he now understands it more deeply and no longer suffers from that problem. If that shakiness, in turn, had been the last *reason* why Cyc had given Elle's model this bug, then Cyc would allow Elle to have a mini-epiphany, report such to the user, and from now on[13] not have this bug in the Elle model any longer, not make this mistake in the future.[14] That improved Elle model concludes that Elle should

[13] So long as this user doesn't relapse into this bug; more precisely, in those future contexts where the user model BELLA ascribes to Barry lacks that particular shakiness assertion.

[14] Generally this is not instantaneous and permanent learning on the part of the user, hence all that changes is a decrease in the frequency with which Elle will make this mistake. Gradually, eventually, she won't make it any more. Over an even longer time frame, some rules might predict that Elle may begin to forget this, or she may do a superficially similar type of problem where rules predict cognitive dissonance occurring; in either of these bases, that bug could recur in Elle's mental model, hence the frequency of this sort of error might increase again.

not have subtracted 3 from both sides of the equation in the previous step, and a message gets sent to the math panel: "I see what I did wrong. Let's go back to the 3x=7 V step." At this point, the intervening Step42 step is grayed out, the 3x= 7 V step (Step41) is copied down as if it were a brand new step, and given a new ID number, say Step45. This does not cause an infinite loop, because Elle's previous bad model has been replaced by a less-bad one, and in particular Elle will not repeat the mistake of subtracting 3 from each side of the "3x=7" equation. Instead, she will proceed to apply her phase 1 simplifying rules (no simplifying to do), her phase 2 isolating rules (no isolating to do), and then—correctly!—her scaling rules (yes, scale by dividing both sides by 3).

Case 3. A poor "correction" of Elle. Let's say the user selected option [id 102] instead of [id 100] or [id101] or [id103], at Step42, above. Elle turns the equation into $\frac{3x}{7} = \frac{7\,\text{volts}}{7}$, and then simplifies the right hand side. This is not wrong, exactly, just an unnecessary complication. If user selects this, the new user model will indicate (if it doesn't already) that Barry does not understand why it's better to turn the *coefficient of x* into "1" than to turn the *non-x term* into "1". Elle might get stuck here, depending on her model, and back up to the Step42 choice again, or she might know that to convert the left side of "$\frac{3x}{7} = 1\,\text{volt}$" into x by multiplying both sides by 7/3, or (even more obscurely) *dividing* both sides by 3/7. Even if these do eventually get Elle to a correct solution, the path is much longer and more awkward than necessary.

Case 4. A very direct, very heavy-handed correction of Elle. Let's say the user selected option [id 103]—"**Should we just divide both sides by 3 instead?**"—instead of [id100] or [id101] back at Step42. That is slightly better skill-wise than the other options, because it requires fewer reasoning steps on Elle's part—hence it's slightly better (for a human tutor to make to a human tutee, and slightly better for the user to make to Elle) if and only if the goal is to improve the tutee's *skill*; it is slightly *worse* from the point of view of getting the tutee to understand the mathematics conceptually. And from Cyc's point of view, it is a slightly less clear demonstration that Barry really does understand the underlying math (than if he had selected option id100 or 101). As with option [id100] and [id101], this selection by Barry is evidence—albeit weaker evidence—that Barry's user model should be upgraded if it currently lists him as being shaky on this type of problem. In this case, and especially in the [id100]a nd [id101] cases, Barry's choice may in turn cause Cyc to upgrade the Elle model so Elle no longer makes this sort of mistake, or at least does so less frequently.

After selecting [id103], Elle writes on the Math panel $\frac{3x}{3} = \frac{7\,\text{volts}}{3}$ prefaced by the Cyc NLG-generated comment "Okay, I divide each side by 3." Elle similarly writes, after that equation, "I need to get rid of the coefficient of 3 in front of the x. If I divide both sides of the equation by the same number they will stay equal. So I am going to divide both sides by three, since 3x/3 = x".

One way or the other, then these cases will eventually, usually, hopefully,… lead Elle to the end of this problem. Cyc tells the UI to display the answer: "x=2 1/3 volts" in the Math Panel, along with a chat comment "Good, we're done!" followed by "Let me double-check that." followed by "If x=2 1/3 volts, then 3x is 6 3/3 V, which is 7 V. Good." Followed by "So each small battery is 2 1/3 volts." None of this sequence of messages is interruptible, they are just kept on different lines to make them more readable. Control passes back to the Game, the User goes back to the 228v door

Continuing on (finally!) to interaction Step46, Elle says: "I need to take batteries from here to the next room. How many 2 1/3 volt batteries do I need so the door will open and close without getting stuck?" followed by "Ten 20 V batteries are already stuck there, so this is the equation I should solve:" followed by the equation **"(10 * 20 V)+y * 2 1/3 volts=228 V",** and the comment "I need to simplify and get the y term by itself on the left side of the equation." This proceeds much like the previous linear equation solving, and Elle ends up eventually with **y=12 V**. Since there were 5 small batteries already in the next room, Elle concludes that she must carry at least 12 - 5=7 more from here to there. In one version of the system, Elle comments that she can't carry seven small batteries in her hands, finds a basket that will hold them, takes them to the next room, and hooks them in, along with the five already there, and the door opens.

Open Issues, Future Development and Testing

The above introduction to BELLA raises many questions: What will prevent student brute-forcing and cheating? What about the value of drill & practice? Also, since we aim to begin "alpha-testing" of the latest version of BELLA shortly in schools, that in itself raises a set of additional issues: What resources will the teacher have? How will appropriate students who will benefit from this experience be selected? How should each such student's user model get initialized? Here we provide at least preliminary or expected answers to such questions, and discuss more generally the path forward.

What About Brute-Forcing? By the way, Barry could have had Elle solve this problem by trial and error, in the game: by just filling up the basket with dozens of small batteries, carrying them up the corridor from the 227-V-door room to the 228-V-door room, and adding them one by one until the door unlocks. If the user had Elle do this, then the user model would be updated to indicate that they tend to brute-force problems more often than had been thought, and the next linear equation problem might result in Cyc choosing larger coefficients, to dissuade Barry from solving such problems via guess-and-check. Using a pop-up box to type in the number of batteries to connect in series means that an answer of 1,240 would be fine, but stolidly typing in all the numbers from 1 to 1,240 as separate guesses would reinforce the advantage of—at least sometimes—setting things up as a math problem and "doing the math". Other incentives to "do the math" include (i) having the problem reset each time (in this case, that could be arranged by having all the batteries burn out, in case of a wrong wiring solution, hence new ones would need to be gathered from around the ship), (ii) having a competitor non-player character who will solve the problem and get the reward instead of the user if too much time is taken, (iii) having a points system, a declining energy level, time limits, or other resource limitations that incent efficient, correct solving of the problems.

What About Cheating? Players inevitably circulate "walk-throughs" and post them online, but almost all of Elle's problems can be automatically altered by Cyc, and the order of feedback options randomized, and of course one of the main points of BELLA is that each user should get an experience which is customized to their exact starting state and micro-response history as they progress through it. Players also abuse Save&Load Game features, so while teachers and researchers will be able to have unfettered access to those features, each "real" player will only be able to move forward in the game, what in online games is often referred to as "Iron Man mode": they can save their game, and come back to it, and continue playing, but never be able to go back and reload that earlier save-game.

What About Drill and Practice? Although we want the user to be immersed in the game, and engaged by it, there is also something to be said for the value of their doing a large number of problems (in total, and per unit time), not just a few very well motivated ones. We have some in-game systems in place already, and some which are designed but are still being implemented, to make this problem-density adjustable, including (i) problems which enable resources to be gathered, such as mining ore deposits on the planet's surface, can "feed" smelting and refining equipment which feeds reactors and engines, etc. which keep needing more and more fuel, and the varied locations of the ore deposits and number of deposits provide a limitless source of organically-motivated problems; (ii) items (e.g., electric drills) in the game can wear out over time, which forces the player to go back and repeat some problems similar to ones done earlier, in order to build replacement items; (iii) Elle can be made to forget things, forcing the user to re-tutor her by solving problems similar to ones she'd already solved; (iv) many problems can be converted from one-step to two- three- or more steps, such as fueling a vehicle where the fuel itself can be created only through mixing components and each component can be obtained only through sub-component mixing or assembling devices to produce those components; (v) replaying the whole game on a harder "difficulty level", by having an Elle that's harder to teach, having problems with larger and more complicated numbers, shorter timers, and so on.

How Should a New User's Model be Initialized? 5w?>We have several candidate approaches and expect some combination of them will suffice: (i) Pretesting; our current pretest involves a set of math questions where the candidate user is asked not to *solve* them but to rate them on a 1–5 scale of "easy" to "no idea", and a set of personality-testing questions to determine their empathy and patience levels. (ii) Interviewing the teacher about this student, essentially having the teacher fill out the pretest for the student; (iii) Starting with a normative model of the typical sixth grade math student who benefits from BELLA; (iv) Ingesting user data from other math learning applications; and (v) Starting with a very low-ranked user model in all 150–200 areas, and rapidly increasing the appropriate user model components early on in the game. One potential benefit of that last approach is that it will provide a natural way to teach students how to use the interface on fairly simple problems—Elle would be making glaring errors initially—and also start them off with a positive experience that builds self-efficacy and self-esteem as they quickly notice and correct those obvious mistakes.

How Can We Incent Deep Understanding and Disincent Memorized Algorithms? In observing pairs of humans playing "Clued In" (Fig. 1), we often saw even the "A" students exhibiting very rigid behavior: If the tutee deviated at all from the way that the tutor would have solved the problem, the tutor would stop them, say they were wrong, and tell them what to do. It became clear that even these good students had memorized algorithms and skills, much like magic incantations, and either never had the underlying conceptual understanding or had since forgotten some of it. To address this issue, we need to model the way that the user approaches and solves a kind of problem, and then have Elle sometimes intentionally solve it a different way, to stretch the user's understanding; or get confused at a step where the user model predicts that user is operating on memorized skills alone. We or other researchers will need to create the rules to more precisely decide the occurrence of these variations.

What About Forgetting and Cognitive Dissonance? An obvious issue which we have only started to address is *forgetting,* or to put it more positively *timely reinforcement.* We all know that when students are taught concepts and procedures in a "unit" or block, forgetting occurs unless they have multiple delayed opportunities to use those concepts and procedures. The logarithmic forgetting curve was well understood over a century ago: Ebbinghaus (1913) popularized it and showed that it can be remediated by a handful of refreshing just before they would have become *necessary* (Fig. 8). We are still tuning Elle's forgetting algorithm, such that she will lose some knowledge with disuse, and require help from the student on material previously covered. In-game, this can emerge from utility devices which gradually wear out, or more prosaically by just having Elle forget things (ideally just *prior* to when the user would have forgotten them.) This is exacerbated when superficially similar concepts are involved: students who used to multiply fractions efficiently get confused by addition of fractions and start putting fractions over common denominators before multiplying them, and then don't multiply the denominators; or conversely add fractions by adding their numerators and

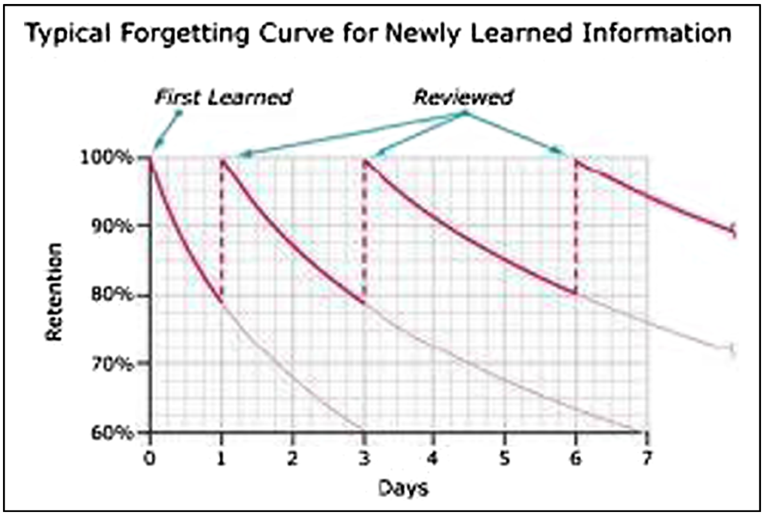


Fig. 8 Ebbinghaus's well-known forgetting versus reminding curve. From: http://www.learningspy.co.uk/featured/deliberately-difficult-focussing-on-learning-rather-than-progress/

adding their denominators. That sort of cognitive dissonance may be ameliorated by continually mixing the *types* of problems, not leaving them sorted into a sequence of curriculum "units", and BELLA could do that same sort of intermixing of problem types to keep the user mindful of recognizing the type of problem they were about to solve rather than falling back on syntactic or, even worse, temporal cues.

Teacher and Administrator Interfaces and Utilities We have begun to evaluate BELLA starting with small scale user tests where we can conduct one-on-one observations, but larger, classroom-scale trials will soon follow. Prior to getting to that point, we will need to understand and incorporate what teachers want from BELLA with regard to providing information about their students. We will need to understand how teachers want to monitor student progress and provide them with both individual and aggregate views of class performance. Our initial implementation of such interfaces and tools can be seen in Figs. 4c, d, and 5, but much empirical work remains to be done in this area.

Student Interfaces and Utilities What should the student be able to see, besides the game itself? Should they able to see the current Elle model (probably so), and if so in what form? Other learning-by-teaching systems provide this capability (e.g., Betty's Brain and MCLS); however, it is not clear whether such specific and frank information about (BELLA's model of) his/her abilities, competencies, inabilities, and incompetencies would be of benefit, and possibly even harm, to the BELLA user. It may be motivational for students to be able to view their progress in the form of a score card of concepts and procedures that that have been covered in the course of the storyline and that they have helped Elle master, or an aggregation of that such as Elle's test results and report card. This needs to be studied empirically. Should they be able to drill down further in asking her why she did something, why she did *that,* etc.? Should they be able to see how their responses and inactions impact their model? Impact Elle's model? These are all questions which can and should be studied anecdotally at first and then using more rigorous educational research methods.

How to Motivate the Users We also intend to explore multiple methods for keeping students engaged. Through gamification, obviously, we aim to provide a steady stream of incentives. As the user "levels up" they should gain new types of interaction options (in addition to other rewards). The gradual introduction of new options should allow bypassing the overwhelming complexity students experienced with our initial user interface (in which all possibilities were present from the start). Some actions that require multiple steps in an early problem may later be chunked as a single macro-action or shortcut. As the story progresses, these methods will hopefully increase the user's sense of responsibility for Elle and rapport with Elle, and through those the overall motivation and overall effectiveness. Looking farther out in time, we hope that we and/or others will experiment with extending this approach to other grade levels and other subjects. But some users are motivated by constructing, and they should be able to craft new math problems for other future users. Still other users will be much more effectively motivated by competition, and there could be mathletic tournaments where different tutored Elle's compete against each other within a class, across classes, or even across schools as in science fairs. Some users will care about certificates and "high score" bragging rights. Some users will be motivated more by tiny but real-world prizes such as candy bars or $1 app downloads. So another direction for future development is to create an effective pretest to decide what will motivate this particular user, and provide that differentially for them.

Does it Work? It almost goes without saying that a key performance indicator is the effectiveness of this pseudo-LBT puppeteering paradigm. Short and long term studies should measure its absolute benefit and compare that with the use of other sorts of teachable agents in education and, more generally, as compared with the effectiveness of all the other paradigms of educating students. If some subpopulation of students does particularly benefit from our approach, devising diagnostic tests to identify that sub-population will be an important area for future research. This dovetails with the pretesting to initialize the user model: extreme ranges of scores on that test might signify “They are not ready for BELLA yet”, or “It would be too easy for them”, or “They won’t be motivated to play it”, or “They won’t benefit much from playing it for some other, more subtle reasons.”

In conclusion, we hope that BELLA will be a useful research tool to study aspects of how learners interact with it, and through that explore questions about how people learn.

Acknowledgments We would like to thank Alan Kay and the staff members at Cycorp who built and experimented with BELLA during Year 1, including David Ayliff, David Baxter, Andrew Beck, Brad Bouldin, Jon Curtis, Gloria Dholakia, Keith Goolsbey, Jessica Gutierrez, John Jantos, Tim Jones, Saket Joshi, Robery Kahlert, Babak Kaveh, Louis Lupin, Emma Pallen, Sheldon Picatti, Karen Pittman, Dan Quattrone, Ben Rode, Dave Schneider, Abhishek Sharma, Blake Shepard. We also like to acknowledge the constructive reviews provided on an earlier version of this paper. This material is based upon work supported by the US Army ACC-APG Natick Contracting Division under Contract No. W911QY-12-C-0106,[15] and we wish to acknowledge our research funders, the Advanced Distributed Learning Initiative and, more generally, the Office of the Secretary of Defense of the United States, and the individuals there and at DoDEA who have helped with BELLA design and feedback, including Marcus Birtwhistle, Rob Chadwick, J. Dexter Fletcher, Steve Hicks, Patrick Martin, Kristy Murray, Jonathan Poltrack, Elaine Raybourn, Damon Regan, and Frank DiGiovanni.

Appendices

Appendix 1. Detailed Lead-In to the Example Problem Scenario

This is a preface to the Example Scenario, above. It comprises mostly in-game actions, and is not relevant to the main theme of this paper, but we include it here for readers who better want to understand the context in which the Example Scenario, above, occurs.

At the first (and simplest) door, there is a control panel right at the door, a label “60 V”, a crate of large unmarked batteries, and a place next to the door to hook them in. The player (the user) can click on the control panel at the door which directs Elle to walk over to it. Elle pushes a button on the control panel. Nothing happens. The Game opens a chat window, and sends a message to the Cyc webservice that requests

[15] Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the US Army ACC-APG Natick Contracting Division.

incidental dialog; this message is of the form <incidental-dialog?ZONE3-PANEL0 user-id interaction-id timestamp>. Cyc looks up relevant incidental dialog and sends the string “This isn’t working.” to the Game. Elle says in the chat window: “This isn’t working.” Similarly, the player could click on the door, and get the message “The ship’s main power is out. This door needs power to open.” The player can click on the crate of batteries, and Elle goes over and picks it up. (The player can click on other things, move Elle around, etc. but that won’t open this door. There is another door, which requires a combination to open, and there is a clue on the ground to what that combination is.) The player clicks on the place next to the door where any number of batteries would fit, and Elle hooks up that battery. This repeats two more times. After three batteries are in place, the door hums slightly. The user can click on it and Elle goes over to it and is able to push it open now. This simple problem introduces the user to the various elements of this genre of door-opening challenges. Since three big batteries in sequence totaled to 60 V, it also forms a simple tutorial for how to help coach Elle as she works through math problems, in this case $3x=60$ V.

Door#1 opens onto a corridor leading to the second door, and this one is not so easy to get past. It is obviously much larger and heavier than door#1 was. Elle and the user see its label, “227 V.” They also see a nearby crate of large batteries each now marked “20 V” (if Elle and the user together correctly solved the Door#1 math problem $3x=60$ V), and a crate of small batteries whose voltage is not marked (but these appear to be the same as each other). 227 and 20 (and the voltage of the small batteries, and the voltage of the next door) are zone parameters—game variables the values of which are selected by BELLA based on the user model. E.g., if the current user model says that the user’s comfort zone on dividing integers of this size is low, then the numbers chosen by BELLA might be smaller. If the current user model says that the user tends to “brute force” their way through problems by guess-and-check, then the numbers chosen by BELLA might be larger, to dissuade them from doing that here. BELLA also applies rules that constrain the choices to forestall its accidentally choosing variable values that admit easy solutions such as 240 and 20, in which case no small batteries at all would need to be used, in order to open door#2.

This is a good lead-in for the reader to go through the detailed Example Scenario. It illustrates how setting up and solving linear equations can occur naturally in the course of game-play. The next Appendix gives a few more examples of how sixth grade math problems arise naturally in BELLA.

Appendix 2. More Examples of Game-Emergent Math Problems

As of the time of this writing, there are about 150 different problems which can arise in the game, and which Elle can solve with the user’s tutorage. We are also developing a new problem crafting interface, which should accelerate the rate of creation of new problems, and allow teachers and students to propose new ones to add to the game themselves. Here are some examples illustrating the way that various sorts of sixth grade Common Core Curriculum math problems can arise in BELLA.

1. Elle can launch mining robots from the ship, but hasn’t yet figured out how to control where they land. The direction is random, the distance is random, but their maximum range is 10 miles. Elle needs about ten of these mining robots to land in

a particular 2×2-mile square area. She has 100 robots to launch. Should she (a) launch them, this will probably work, (b) skip it, this is very unlikely to be enough robots? (Answer: she should calculate it will take about 250 random launches, so this is not worth her time.) If they do the math first, they will realize they just saved themselves ten minutes of mindless button-pushing. Or, if the user sits there for 10 min and pushes the launch button 100 times, Elle will go through the math herself and see that it was doomed to failure; that will hopefully reinforce their belief that sometimes it is faster to "do the math" than to "just hope for the best".

2. Later on, Elle finds one single advanced mining robot which she will be able to command, once it lands; it lands at some coordinates (x,y), i.e., x units east and y units north of the ship. She sees a sensor display of the location of ten specific resource sites the rover needs to visit. But these are given in coordinates based on the ship being (0,0), and the rover is unfortunately commanded by telling it *relative to wherever it currently is* how many units to go in the x (east) and y (north) directions before digging again. Elle needs to find a reasonable route for it, and then communicate that route as a series of relative x,y movements, watching it on-screen as it moves around, seeing it and the deposit sites it needs to visit. There is the appearance of pressure on Elle in the form of the robot's fuel gauge, for the first such foray there will be just enough fuel no matter how circuitous the route Elle directs it along.
3. Elle encounters color-coded doors with keypad locks. Besides the usual calculator arrangement of keys, there is an Input and Output LED display. One of the security consoles on the ship's bridge contains formulae for the lowest-security doors; generally these will be linear algebraic expressions in terms of the Input x, e.g., "20 - 3*x*". The first few doors will be simple enough that Elle and the user could just stand there and try one natural number after another, and get through in a minute. Some of these panels provide a "too high" or "too low" readout, that even facilitates something like binary search guessing. In some cases, the formula is not written out already for the student, but they can put in any IN value other than the one asked for (in this case, 21) and the door will flash what the OUT value would have been for that input. Gradually, the doors' formulae get more complicated. With increasing frequency, the doors provide the output and need the user to type in the input on the keypad, so the user needs to invert the formula, to find the value of *x* which, using the security console formula for that color of door, would result in that given output. We have recently added a few even more difficult keypad-locked doors, e.g., to optional equipment rooms, and we have made these problems more *emergent* by, e.g., making the formulae and its elements found by having Elle search cabins, count things on the ship, and access real facts about the Earth (facts which are unlikely to change on Earth between now and 200 years from now, such as the number of continents.)
4. Elle needs to power a ship's system whose fuel supplies are gone. However, the ship had an extensive laboratory where workers could synthesize various chemicals, including fuel for the system in question. Enough of the lab is intact for her to synthesize the fuel herself. Before she can make the attempt, Elle must first discover the recipe for the fuel. It in turn is made of two substances (thiotimoline and hydrosilium) mixed in a particular ratio, and each of those two substances needs to be synthesized separately out of other substances. Elle needs to

figure out where the component substances are, how much of each she will need, move them to the mixing apparatus, do the synthesis of the thiotimoline and hydrosilium and then mix those. There are many opportunities for errors, and for diagnosing and fixing those errors, and not all the errors require restarting from scratch. E.g., if Elle makes too much of some component, she and the user can learn from that, can realize that they just wasted some time and resources.

5. Elle, still in the enormous crashed starship, sees a sensor display of some of the immense tunnel system underneath the surrounding terrain, and sees the location of an unusual site to investigate. The length of each point-to-point tunnel segment is marked (including "the ship" which crashed near one of the tunnel openings), but Elle has to do the math herself to figure out which will be the shortest total path to get to the unusual site. She has a motive for minimizing her time off-ship (a competitor/opponent who will simultaneously be racing to get to the same site). This is a family of increasingly (or, if the user has trouble with it, decreasingly) difficult problems, introducing complications such as the speed that Elle can go in different tunnel lighting conditions, roughness of terrain, and steepness of incline. It also deeply conveys the "three meanings" of the relationship between speed, distance, and time.
6. Given a metric nut and a large assortment of non-metric wrenches (whose sizes are expressed as maximally simplified fractions and mixed numbers, as in real life), select the closest non-metric wrench for the job. Elle and the user can see, each time a wrench is tried, if it's too large or too small.
7. Elle needs to know how many crates of batteries she has, total, on board the ship. They are all stacked up in a huge rectangular fashion $L \times W \times H$. But one of the top corners is missing; it's where the battery crates must have been taken from, already. The "missing corner" has dimensions l, w, h. (Elle needs to set this up as $L \times W \times H - l \times w \times h$, and/or needs to learn from doing it *in*efficiently.)

References

Aleven, V., McLaren, B., Roll, I., & Koedinger, K. (2006). Toward meta-cognitive tutoring: a model of help-seeking with a cognitive tutor. *International Journal of Artificial Intelligence in Education, 16*, 101–130.

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

ATLT (Advanced Training and Learning Technologies). (2012). *Game-based tutoring technologies final report*. Virginia Beach: ATLT.

Berliner, D. (1989). Being the teacher helps students learn. *Instructor, 98*, 12–13.

Biswas, G., Schwartz, D. L., Bransford, J. D., & The Teachable Agents Group at Vanderbilt. (2001). Technology support for complex problem solving: From SAD environments to AI. In K. Forbus & P. Feltovich (Eds.), *Smart machines in education* (pp. 71–98). Menlo Park: AAAI/MIT Press.

Blair, K., Schwartz, D., Biswas, G., & Leelawong, K. (2006). Pedagogical agents for learning by teaching: teachable agents. *Journal of Educational Technology and Society*, Special Issue on Pedagogical Agents.

Brizuela, B. M., Martinez, M. V., & Cayton-Hodges, G. A. (2013). The impact of early algebra: results from a longitudinal intervention. *Journal for Research in Mathematics Education, 2*(2), 209–241.

Brophy, S., Biswas, G., Katzlberger, T., Bransford, J., & Schwartz, D. (1999). Teachable agents: Combining insights from learning theory and computer science. In S. P. Lajoie & M. Vivet (Eds.), *Artificial intelligence in education* (pp. 21–28). Amsterdam: Ios Press.

Brown, J. S., & VanLehn, K. (1980). Repair theory: a generative theory of bugs in procedural skills. *Cognitive Science, 4*, 379–426.

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

Chin, D. B., Dohmen, I. M., Cheng, B. H., Oppezzo, M. A., Chase, C. C., & Schwartz, D. L. (2010). Preparing students for future learning with teachable agents. *Educational Technology Research and Development, 58*, 649–670.

Cohen, P. A., Kulik, J. A., & Kulik, C. L. C. (1982). Education outcomes of tutoring: a meta-analysis of findings. *American Educational Research Journal, 19*, 237–248.

Ebbinghaus, H. (1913). *Memory: A contribution to experimental psychology* (trans: Ruger, H. & Bussenius, C.). New York, NY: Teachers College, Columbia University. (Original work published in 1885).

Feurzeig, W., Papert, S., Bloom, M., Grant, R., & Solomon, C. (1969). *Programming-Languages as a conceptual framework for teaching mathematics, Final Report on NSF-C project 558*. Washington, DC: National Science Foundation, Office of Computing Activities.

Fletcher, J. D. (2011). *DARPA education dominance program: April 2010 and November 2010 digital tutor assessments, IDA NS D-4260, Log: H 11-000117*. Alexandria: Institute for Defense Analyses.

Forbus, K. (2012). How minds will be built. *Advances in Cognitive Systems, 1*, 47–58.

Fry, E. (1960). Teaching machine dichotomy: Skinner vs. Pressey. *Psychological Reports, 6*, 11–14.

Graef, R., & Preller, R. D. (1994). *Lernen durch Lehren*. Rimbach, Germany: Verlag im Wald, Rimbach.

Graesser, A. C., McNamara, D. S., & VanLehn, K. (2005). Scaffolding deep comprehension strategies through Point and Query, AutoTutor, and iSTART. *Educational Psychologist, 40*, 225–234.

Gulz, A., Haake, M., & Sivervarg, A. (2011). Extending a teachable agent with social conversation modules – Effects on student experience and learning. In G. Biswas, S. Bull, J. Kay, & A. Mitrovic (Eds.), *Artificial intelligence in education, LNAI 6738* (pp. 106–114). Berlin: Springer.

Husain, L., Meletli, N., Boström, M., Axelsson, K., & Samvik, R. (2010). *Teachable agent for Elevdata's Matematikhuset*. Sweden: Lund University.

Katzlberger, T. (2005). *Learning by teaching agents* (Doctoral thesis). Nashville, TN: Vanderbilt University

Kiili, K., Koivisto, A., Finn, E., & Ketamo, H. (2012). *Proceedings from the European conference on games based learning: Measuring user experience in tablet based educational game*. Sonning Common: Academic Conferences and Publishing International.

Lancaster, J. (1821). *The Lancasterian system of education*. Baltimore: Lancaster School.

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

Lenat, D., & Guha, R. V. (1990). *Building large knowledge-based systems: Representations and inference in the Cyc project*. Boston: Addison-Wesley.

Lenat, D., Prakash, M., & Shepherd, M. (1985). CYC: using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *AI Magazine, 6*(4), 65–85.

Lenat, D., Witbrock, M., Baxter, D., Blackstone, E., Deaton, C., Schneider, D., Scott, J., & Shepard, B. (2010). Harnessing Cyc to answer clinical researchers' ad hoc queries. *AI Magazine, 31*(3).

Li, N., Cohen, W., Koedinger, K. R., & Matsuda, N. (2011). Proceedings from the Fourth International Conference on Educational Data Mining. In M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero, & J. Stamper (Eds.), *A machine learning approach for automatic student model discovery.* Eindhoven: TU/e printservice.

Matsuda, N., Cohen, W. W., Koedinger, K. R., Stylianides, G., Keiser, V., & Raizada, R. (2010). Proceedings from the International Workshop on Adaptation and Personalization in E- B/Learning using Pedagogic Conversational Agents (APLeC). In D. Perez-Marin, I. Pascual-Nieto, and S. Ball (Eds.), *Tuning cognitive tutors into a platform for learning-by-teaching with SimStudent technology.* Retrieved from http://ceur-ws.org/Vol-587/paper4.pdf.

Matsuda, N., Yarzebinski, E., Keiser, V., Raizada, R., Stylianides, G., Cohen, W. W., Stylianides, G., & Koedinger, K. R. (2012a). Proceedings from international conference on intelligent tutoring systems. In S. Cerri & W. Clancey (Eds.), *Motivational factors for learning-by-teaching: The effect of a competitive game show in a virtual peer-learning environment.* Berlin: Springer.

Matsuda, N., Yarzebinski, E., Keiser, V., Raizada, R., William, W. C., Stylianides, G., & Koedinger, K. R. (2012b). Proceedings from the annual conference of the cognitive science society. In N. Miyake, D. Peebles, & R. P. Cooper (Eds.), *Shallow learning as a pathway for successful learning both for tutors and tutees*. Austin: Cognitive Science Society.

Melis, E., & Siekmann, J. (1999). Knowledge-based proof planning. *Artificial Intelligence Journal, 115*(1), 65–105.

Michie, D., Paterson, A., & Hayes-Michie, J. (1989). Learning-by-teaching. In H. Jaakkola & S. Linnainmaa (Eds.), *2nd Scandinavian conference on artificial intelligence 89* (pp. 307–331). Amsterdam: Ios Press.

Millis, K., Forsyth, C., Butler, H., Wallace, P., Graesser, A. C., & Halpern, D. (2011). Operation ARIES! A serious game for teaching scientific inquiry. In M. Ma, A. Oikonomou, & J. Lakhmi (Eds.), *Serious games and edutainment applications* (pp. 169–196). London: Springer.

Nichols, D. (1994). Proceedings of the East-West International Conference on Computer Technologies in Education (EW-ED'94), volume 1. In P. Brusilovsky, S. Dikareva, J. Greer, & V. Petrushin (Eds.), *Issues in designing learning-by-teaching systems* (pp. 176–181). Crimea, Ukranine.

Pareto, L., Haake, M., Lindström, P., Sjödén, B., & Gulz, A. (2012). A teachable-agent-based game affording collaboration and competition: evaluating math comprehension and motivation. *Educational Technology Research and Development, 60*, 723–751.

President's Council of Advisors on Science and Technology. (2010). *Prepare and inspire: K-12 education in Science, Technology, Engineering, and Math (STEM) for America's future*. Washington, DC. Retrieved from http://www.whitehouse.gov/sites/default/files/microsites/ostp/pcast-stemed-report.pdf.

Roscoe, R. D., & Chi, M. T. H. (2007). Understanding tutor learning: knowledge-building and knowledge-telling in peer tutors' explanations and questions. *Review of Educational Research, 77*, 534–574.

Roscoe, R. D., Segedy, J., Sulcer, B., Jeong, H., & Biswas, G. (2013). Shallow strategy development in a teachable agent environment designed to support self-regulated learning. *Computers and Education, 62*, 286–297.

Segedy, J. R., Kinnebrew, J. S., & Biswas, G. (2013). The effect of contextualized conversational feedback in a complex open-ended learning environment. *Educational Technology Research and Development, 61*, 71–89.

Sowa, J. (1995). *Syntax, semantics, and pragmatics of contexts, AAAI technical report FS-95-02*. Vestal: American Association of Artificial Intelligence.

Stern, N. (2011). *Teach some, learn some: The benefits of tutoring*. Retrieved from http://www.brassmagazine.com/article/teach-some-learn-some-benefits-tutoring.

Stone, J. R. (2004). *The Routledge dictionary of latin quotations: The Illiterati's guide to Latin maxims, mottoes, proverbs, and sayings*. New York: Routledge.

Tschurenev, J. (2008). Diffusing useful knowledge: the monitorial system of education in Madras, London and Bengal, 1789-1840. *Paedagogica Historica: International Journal of the History of Education, 44*(3), 245–264.

Uresti, J., & du Boulay, B. (2004). Expertise, motivation, and teaching in learning companion systems. *International Journal of Artificial Intelligence in Education, 14*, 67–106.

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

VanLehn, K., Freedman, R., Jordan, P., Murray, C., Osan, R., Ringenberg, M., Rose, C., Schulze, K., Shelby, R., Treacy, D., Weinstein, A., & Wintersgill, M. (2000). Proceedings from intelligent tutoring systems: 5th international conference: Vol. 1839. Lecture notes in computer science. In G. Gauthier, C. Frasson, & K. VanLehn (Eds.), *Fading and deepening: The next steps for Andes and other model-tracing tutors*. Berlin: Springer.

Wagster, J., Tan, J., Wu, Y., Biswas, G., & Schwartz, D. L. (2007). Proceeding from the 29th meeting of the Cognitive Science Society. In D. S. McNamara & J. G. Trafton (Eds.), *Do learning-by-teaching environments with metacognitive support help students develop better learning behaviors?* Nashville: Cognitive Science Society.

Walker, E., Rummel, N., Walker, S., & Koedinger, K. R. (2012). Noticing relevant feedback improves learning in an intelligent tutoring system for peer tutoring. In S. A. Cerri, W. J. Clancey, G. Papdourakis, & K. Panourgia (Eds.), *Intelligent tutoring systems, 2012* (pp. 222–232). Berlin: Springer.

Winston, P. H. (1970). *Learning structural descriptions from examples*. (doctoral thesis). Cambridge, MA: MIT.

Zhao, G., Ailiya, & Shen, Z. (2012). Learning-by-teaching: designing teachable agents with intrinsic motivation. *Educational Technology and Society, 15*, 62–74.