Interpreting Freeform Equation Solving

Anna N. Rafferty and Thomas L. Griffiths

[1] Computer Science Department, Carleton College, Northfield, MN 55057 USA, arafferty@carleton.edu
[2] Department of Psychology, University of California, Berkeley, CA 94720 USA, tom_griffiths@berkeley.edu

Abstract. Learners' step-by-step solutions can offer insight into their misunderstandings. Because of the difficulty of automatically interpreting freeform solutions, educational technologies often structure problem solving into particular patterns. Hypothesizing that structured interfaces may frustrate some learners, we conducted an experiment comparing two interfaces for solving equations: one requires users to enter steps in an efficient sequence and insists each step be mathematically correct before the user can continue, and the other allows users to enter any steps they would like. We find that practicing equation solving in either interface was associated with improved scores on a multiple choice assessment, but that users who had the freedom to make mistakes were more satisfied with the interface. In order to make inferences from these more freeform data, we develop a Bayesian inverse planning algorithm for diagnosing algebra understanding that interprets individual equation solving steps and places no restrictions on the ordering or correctness of steps. This algorithms draws inferences and exhibits similar confidence based on data from either interface. Our work shows that inverse planning can interpret freeform problem solving, and suggests the need to further investigate how structured interfaces affect learners' motivation and engagement.

1 Introduction

Observing students' solutions to problems can provide valuable insights into their understanding. The approach that a student takes may reveal gaps in her knowledge, or the way she executes a particular step may demonstrate a misunderstanding or misconception. Often, educational technologies structure problem solutions in order to provide targeted, step-by-step feedback to learners [8]. This structuring may include requiring students to get each individual step of a problem correct before continuing, to complete steps in a fixed order, or to indicate what action they are taking to go from one step to another.

While this structuring of students' problem solving may assist in making automated inferences about students' understanding and step-by-step feedback could be beneficial for student learning, it could be frustrating to some learners. Entering individual actions may be time consuming, making practice less efficient, and could be infeasible for some beginning learners. Rigid structuring of the ordering of steps or their correctness may lead some users to disengage and give up more easily, potentially outweighing the benefits of immediate feedback.

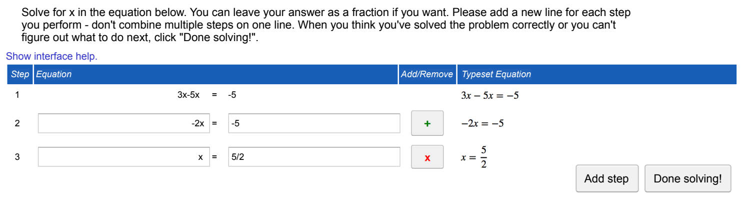
We explore the tension between providing structure to push learners towards correctness and limiting frustration through an experiment comparing user satisfaction and improvement in problem solving across two interfaces. Our experiment focuses on equation solving both because algebra is a core topic in curricula, and because interfaces for this domain could allow very freeform solving. We find users are more satisfied with the freeform interface, and we thus explore how to automatically diagnose understanding based on data from this interface. In the Bayesian inverse planning algorithm we develop, we use the patterns of equation transformations, both correct and incorrect, to compute student ability parameters for distinct skills. We find that the model's inferences are equally confident when interpreting data from either interface. We end by discussing implications and highlighting future directions as well as limitations of this work.

2 Interfaces for Algebra Tutors

A number of systems with a variety of interfaces exist to help students practice multistep problem solving. Because we focus on linear equation solving, we concentrate here on systems for algebra. Some systems ask students to enter only a final answer (e.g., ALEKS [6]), preventing feedback on individual steps. In ASSISTments [15], questions are often structured to ask students first for their final answer, and then, if the student answers incorrectly, scaffolding is provided to structure the problem and divide it into smaller parts, each of which must be answered correctly before the student can continue. Cognitive Tutor Algebra [8] typically structures problem solving into individual steps which students must complete correctly before continuing, and feedback is targeted at these individual steps. This tutor improves students' algebra skills and standardized test scores [8]. Previous work comparing the typical interface to a handwriting interface for this tutor, which only provided feedback on students' final answers, found that while step-by-step feedback was more beneficial for learning [1], students preferred the handwritten interface [2]. Beyond the efficiency of handwriting, students may also have appreciated being able to show their work without constraints on step correctness. These studies point to a divide in students' preferences for particular interfaces, which may affect motivation and engagement, with learning outcomes. Research on novice programmers has found fewer differences in outcomes between learners who received step-by-step feedback versus less intrusive feedback, although students who received less intrusive feedback tended to take longer to complete the same number of exercises [5].

3 Satisfaction with a Freeform vs. Structured Interface

Given the prevalence of step-by-step feedback in existing tutors and previous work demonstrating a potential preference for more freeform responses, we conducted an experiment to explore how the interface which people used to solve


Solve for x in the equation below. You can leave your answer as a fraction if you want. Please add a new line for each step you perform - don't combine multiple steps on one line. When you think you've solved the problem correctly or you can't figure out what to do next, click "Done solving!".

Show interface help.

Step	Equation			Add/Remove	Typeset Equation
1	3x-5x	=	-5		$3x - 5x = -5$
2	-2x	=	-5	+	$-2x = -5$
3	x	=	5/2	x	$x = \frac{5}{2}$

Fig. 1. Screenshot of the step-by-step problem solving interface on the website. The first line in the interface shows the problem the user must solve, and the user adds lines to show her problem solving steps.

algebraic equations affected their satisfaction with the system and their learning outcomes. We focused on adult problem solvers who had some algebra experience, but may not have used their skills recently. The structured interface could be helpful to these participants by giving them more specific feedback while solving equations, but might also be frustrating due to limiting their ability to demonstrate what they do know, as these learners are not complete novices but are in need of remediation. We hypothesize that this frustration will lead these participants to give up more quickly and be less satisfied with the system.

3.1 Methods

Participants. 40 participants in the USA were recruited from Amazon's Mechanical Turk (AMT) and compensated \$3 for session 1, \$5 for session 2, and \$7 for session 3. Participants had experience with algebra, either in a secondary or postsecondary class, and had not completed college math classes beyond algebra.
Stimuli. Participants completed a multiple choice assessment, solved algebra problems on a website, and responded to surveys about their mathematics background and the usability of the website. The multiple choice assessment was based on College Board ACCUPLACER® tests used for math placement in many postsecondary institutions[11]; questions were adapted from sample questions from the College Board[4]. The assessment included 12 elementary algebra questions (the same length as the Elementary Algebra ACCUPLACER), and 16 college math questions (shortened from 20 questions in the College Math ACCUPLACER). Instructions informed participants that if they did not know how to solve the problem, they could leave the question blank.

On the algebra website we developed, participants are shown the equation to solve, and with each problem step, they add a line to show their work (Figure 1). In the *corrected* interface, the current step is checked via a symbolic algebra system [9] when the add step button is clicked. If the step is mathematically incorrect, combines multiple steps, or is not the best next action, then a dialog tells the participant to correct the step before continuing. The dialog distinguishes among syntactic, mathematical, and action choice errors. Allowed action choices are calculated by comparing the user's actions to all possible legal actions. We allow any action that is close to optimal (see Section 5 for details).

In the *freeform* interface, the system checks for syntactic errors when the user submits the entire set of steps for a problem. Rows with syntactic errors are highlighted. After submitting the problem and either correcting syntactic errors or moving forward without correction, any steps with mathematical errors are highlighted and the user is given the opportunity to look at her problem solving. This occurs to provide similar amounts of feedback to participants in both conditions. The user chooses when to move on to the next problem.

In the usability survey, all participants completed ten questions on a 5-point Likert-scale, with participants in the *corrected* condition completing two additional questions. The initial ten questions were adapted from existing usability surveys, such as [10], and focused on perceived learning as well as satisfaction with features of the website. The *corrected*-only questions focused on feelings about having to get each step correct before continuing and whether they found this feedback helpful. The survey also contained three open-ended questions.
Procedure. Participants completed three sessions, separated by at least one day. The first session included the multiple choice assessment followed by the mathematical background survey. In the second session, participants were randomly assigned to one of two conditions and solved 20 problems on the algebra website. The website included a short tutorial about how to use the interface. This tutorial was identical across conditions except for explanations of differing features. In the final session, participants completed the same assessment as in session one, and then responded to the usability survey.

3.2 Results

Overall, performance improved from pretest to posttest. A repeated measures ANOVA with factors for condition and pre- versus post-test showed a small but significant improvement ($F(1, 79) = 14.5$, $p < .001$; mean 11.3 correct on pretest to 12.8 correct at posttest, Cohen's $d = .27$); no interaction between condition and time of test was found ($F(1, 79) = 0.66$, $p = .42$). Given the short nature of the intervention and the far transfer nature of some questions on the assessment assessment, we would not necessarily expect a large improvement. As shown in Figure 2a-b, participants in both conditions attempted similar numbers of problems and gave comparable numbers of correct answers on the website. However, participants in the *corrected* condition entered fewer answers, likely due to being unable to give a correct next step. Note that participants in the *corrected* condition could give an answer without the intervening steps, but they would be warned that they had not entered the correct next step. We counted these cases as having given an answer (and as correct if the answer was correct).

The low scores on the assessment and the website demonstrate that AMT participants were not already proficient with algebra and have gaps in their knowledge and misunderstandings. Comments from participants suggested that their scores were not due to lack of effort but due to the difficulty of the task: "humbling" and "very hard" were common sentiments. Given the need for mathematical skills across a range of careers, developing systems that meet the needs of adult learners and testing effectiveness for these learners is an important goal.

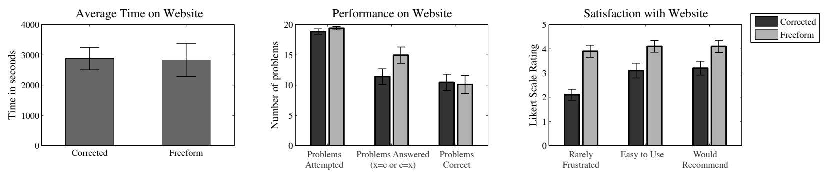


Fig. 2. (a) Time spent on website by condition. Error bars show one standard error. (b) Performance on website by condition. "Attempted" problems are those where the user entered at least one step; correct indicates the final answer was mathematically correct. (c) Website usability ratings.

Satisfaction with the website varied between conditions. As shown in Figure 2c, participants in the *freeform* tended to be less frustrated ($t(38) = 5.3$, $p < .001$) and thought that the website was easier to use ($t(38) = 2.6$, $p < .05$). They also more strongly agreed that they would recommend the site to others ($t(38) = 2.4$, $p < .05$). Responses to other questions, including ability to understand the interface and finding feedback helpful, trended towards more satisfaction with the *freeform* interface but were similar across conditions.

4 Inverse Planning

Given that the *freeform* interface is less frustrating, we would like to use data from this interface to infer understanding. However, most existing models require that highly structured data and these models do not use the pattern of action *choices* to draw conclusions about misunderstandings. We thus developed a Bayesian inverse planning algorithm to model equation solving, enabling us to make specific inferences based on data from either interface.

Inverse planning is a type of inverse reinforcement learning in which an agent's observed actions are used to make inferences about its goals and understanding of the world. Previous applications to cognition have focused on inferring people's goals based on their action choices [3] and recognizing people's beliefs about how their actions affect their environment [14]. Here, we focus on using inverse planning to diagnose (mis)-understandings, where we assume we have some space of hypotheses θ representing all possible understandings. This diagnosis is computed as a posterior distribution over θ given observed problem solutions $d_1, \ldots, d_N$: $p(\theta|d_1, \ldots, d_N) \propto p(\theta) \prod_{i=1}^{N} p(d_i|\theta)$, where each term $p(d_i|\theta)$ is the likelihood of the observed data for a single problem.

To calculate the likelihood for a problem, we must model how people choose actions and how their transitions from one problem solving step to another are affected by their understanding. We model these choices and transitions using a Markov decision process (MDP). MDPs are decision-theoretic models for action planning that are commonly used when multiple, non-deterministic actions must be taken over time. Formally, MDPs are defined by a tuple $\langle S, A, T, R, \gamma \rangle$. At each time step t, the process is in some state $s \in S$. The agent chooses an action $a \in A$, and with probability $p(s'|a, s)$, specified by the transition model

$T(s,a,s')$, the state transitions to next state s'. The reward model $R(s,a,s')$ encodes the incentives. To calculate the likelihood of an observed solution, we use the Markov assumption that given the present, future states are independent of past states:

$$p(s_{1:M}|\theta) = p(s_1|\theta)\prod_{i=1}^{M-1} p(s_{i+1}|\theta, s_i) \propto \prod_{i=1}^{M-1}\sum_{a\in A} p(s_{i+1}|\theta, s_i, a)p(a|\theta, s_i), \quad (1)$$

where the final state s_M is reached by taking a **stop** action. We sum over actions due to the fact that in our data, only steps, not actions, are observed. The term $p(s_{i+1}|\theta, s_i, a)$ can be calculated if we assume that the person's understanding θ maps to a particular MDP: it is the transition model reflecting the person's pattern of moving from one step to another. The term $p(a|\theta, s)$ is the policy for this MDP and is expanded on in the next section as a function of the long term expected value of taking an action a in a state s, known as the Q-value:

$$Q(s,a) = \sum_{s'} p(s'|\theta, s, a)(R(s,a,s') + \gamma \sum_{a'\in A} p(a'|\theta, s')Q(s',a')), \quad (2)$$

where γ reflects the relative weight of immediate versus future rewards. The policy and Q-values can be computed simultaneously through value iteration [17].

5 Modeling Equation Solving

To diagnose understanding from observations of solutions to linear equations, we developed a Bayesian inverse planning algorithm specifically for this domain, building on preliminary previous work [13]. We define the states, actions, and hypotheses for linear equation solving, and then address a technical challenge not present in previous applications of inverse planning to cognition: the state and action spaces for modeling linear equation solving are infinite.

Each equation is mapped to a state, and actions correspond to the way a person changes the equation when solving. The state is represented as the set of terms on each side of the equation, ignoring their ordering. We consider six general types of actions, corresponding to possible equation transformations:

1. **move**: Move a specific term from one side of the equation to another.
2. **divide**: Divide both sides by the coefficient of a specific term.
3. **multiply**: Multiply both sides by a specific constant.
4. **combine**: Combine two specific terms on the same side of the equation.
5. **distribute**: Multiply out a specific complex term like $3(x + 6)$ to get $3x + 18$ or transforming $-3(x + 6)$ to $-(3x + 18)$.
6. **stop**: Stop solving the current problem.

The first five actions represent typical equation solving behavior; below, we describe how errors in these actions are modeled. The final action is taken by a student who believes she has finished solving the problem or is giving up.

The reward model includes a positive reward for choosing **stop** in a state $x = c$ or $c = x$, where x is the variable and c is a constant, and a negative reward for choosing **stop** in any other state. A small negative reward is also incurred for each action: using fewer steps is preferable to using more steps.

5.1 Defining the hypothesis space

We define the hypothesis state θ as a vector representing six different student ability parameters. Four parameters are related to specific actions and are based on mal-rules from prior work [12, 16]. The *sign error* parameter refers to moving a term to the other side without changing the sign ($2x + 3 = 6 \rightarrow 2x = 6 + 3$). The *reciprocal error* parameter refers to multiplying rather than dividing by a coefficient ($5x = 1 \rightarrow x = 5$). The *distributive error* parameter refers to multiplying only the first term in parentheses by a value ($4(x + 3) \rightarrow 4x + 3$). Each parameter is a value in $[0, 1]$ that reflects the probability of producing the error for the relevant action. These first three dimensions of θ thus govern the transition model of an MDP. Representing erroneous rules as probabilities allows us to account for prior data showing inconsistent application of mal-rules [12]. The fourth action parameter indicates whether only like terms may be combined (only constants or only variables). This is implemented by including versions of the *combine terms* action for non-like terms only if this parameter is zero.

Two additional parameters affect each action choice and transformation. The *arithmetic error* parameter is the probability of making an arithmetic error in each operation in a transformation. Like the first three dimensions of θ, this parameter affects transition probabilities. It allows us to distinguish between misunderstandings about the rules of algebra and difficulties with arithmetic. The final parameter relates to solving efficiency. Following prior work modeling human action planning [3, 14], we assume a noisily optimal Boltzmann policy $p(a|s) \propto \exp(\theta_\beta Q(s, a))$, where θ_β controls the noisiness of the policy. Larger θ_β indicates more probability on the actions with highest expected value. Inferring θ_β allows us to detect how efficiently an individual is choosing actions.

5.2 Computing the diagnosis

The diagnosis of understanding is expressed as a posterior distribution over θ. Because θ is continuous, we approximate the posterior using Metropolis Hastings [7]. We place a $\text{Beta}(1, 3)$ prior on the four probability parameters in θ, favoring values closer to normative algebra solving. The prior was not tuned.

Metropolis Hastings calculates the likelihood of the data using sampled values of θ. To calculate the transition model and policy for a specific θ, we must compute the Q-values for the corresponding MDP. We approximate the solution because the state and action spaces are infinite. First, we discretize the actions by creating generic versions of each action that indicate what types of terms (constant, variable, or complex) are acted on but ignore the coefficients. To discretize the state space, we aggregate states into a finite set of possibilities, and then solve $Q(s, a)$ using the aggregated states and discretized actions, such that all equations mapping to the same state have the same Q-values [17].[3]

[3] Q-values are also used to find optimal actions in the *corrected* interface. Assuming no errors and letting $\theta_\beta = 5$, we calculate a close to optimal policy. Actions were accepted if they had probability within $\epsilon = 0.05$ of the highest probability.

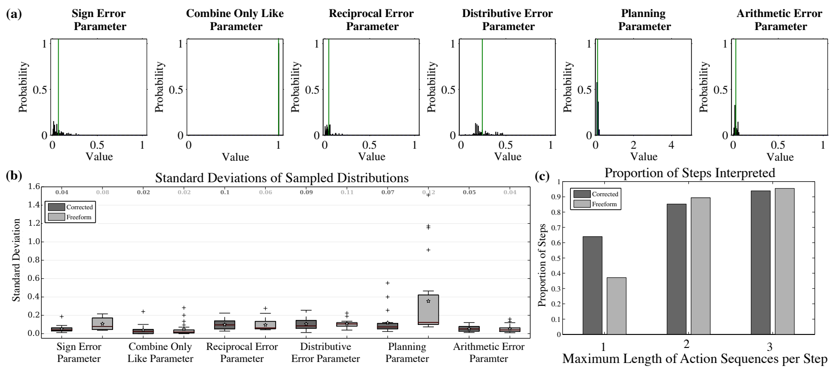


Fig. 3. (a) Diagnosis from a participant who used the *corrected* interface. (b) Distribution of the standard deviations of the sampled distributions, by condition. (c) Proportion of steps where inverse planning found at least one valid action sequence.

6 Modeling Human Problem Solving Data

We applied Bayesian inverse planning to diagnose the understanding of each participant in our experiment. Figure 3a shows a graph of the posterior distributions for a single participant in the *corrected* condition. The somewhat low planning parameter is likely due to the fact that the participant gave up on several problems without reaching an answer. The participant had difficulty with problems involving the distributive property, reflected by the high estimated value for the distributive error parameter. The relatively large spread of the posterior for this parameter suggests that the algorithm has low confidence in an exact value. Since inverse planning computes the diagnosis as a distribution, we can easily see the confidence of the algorithm in particular parameter values.

We explored whether the algorithm's diagnoses were as confident based on the *freeform* data as from the *corrected* data. While interpreting *freeform* data is a natural fit for this algorithm, these data are also likely to exhibit ambiguity. Since this interface does not force particular action choices and allows mathematical errors, there may be cases where more than one action or error could account for a step, and people may combine actions.

To measure how confident the diagnoses were for different conditions, we calculated the standard deviation of the sampled values for each participant. As shown in Figure 3b, the distribution of these standard deviations are generally similar across conditions except for the planning parameter: for most parameters, the lack of constraints in the *freeform* interface did not decrease confidence. For the planning parameter, we believe the difference in confidence is due to participants in the *corrected* condition giving up more frequently. This lead to lower inferred planning parameters on average. Since variance in sampled values tends to increase with the parameter's magnitude, we would expect that this difference in values would result in less absolute confidence for this parameter in the *freeform* condition. This difference also highlights the fact that one might want to separate choices about when to stop solving from suboptimal choices among other actions, which could be incorporated via a two part choice function.

Another measure of the algorithm's performance is its ability to interpret steps as actions in the model. Figure 3c shows the proportion of two-step sequences for which at least one action had non-zero probability; this includes all attempts at steps, including steps that participants entered in the *corrected* condition that were incorrect or the wrong action. When each step transition must be interpreted as a single action, many more steps in the *corrected* condition can be interpreted than in the *freeform* condition. We hypothesized that participants may combine multiple actions in a single step; this could be due to a mismatch between our definition of an individual step and participants beliefs about what defines a step. To address this, we extended the algorithm to sum over sequences of actions rather than only single actions in Equation 1; this may increase noise, as there is more ambiguity, but allows for broader coverage. We only sum over sequences of actions in cases where no single action has non-zero probability. As Figure 2c shows, this results in similar coverage across conditions.

7 Discussion

Our results demonstrate that Bayesian inverse planning can interpret equation solving regardless of which interface is used. However, people who used the *freeform* interface were less frustrated and more able to reach solutions. Several participants commented that they wanted to solve problems "their way"; the *freeform* interface comes much closer than the *corrected* interface to realizing this goal, and inverse planning provides a powerful tool for understanding actions with minimal structuring. While it may be beneficial to eventually guide learners towards more efficient actions, forcing this sequence immediately, especially for those with partial understanding, could be frustrating.

Our results have several limitations. AMT participants may have different motivations than typical algebra students. However, we believe these participants still provide valuable evidence about the usability of the interfaces and have varying algebra skills, similar to other adult learners. Comments from several participants indicated they wanted to learn more algebra or appreciated practicing old skills, suggesting some interest in the topic. An additional limitation is our choice of interfaces. Specifically, some tutoring systems are less constrained than our *corrected* interface, requiring only that steps be mathematically correct but not that they be in a normative order. Our constraint on order represents minimal ambiguity for inverse planning, providing a best case scenario for the algorithm to which we can compare, and the fact that we allowed slightly suboptimal actions means that some action choice was still permitted.

This work demonstrates that inverse planning can interpret equation solving data, providing a way to allow more freeform solutions without sacrificing automated knowledge tracking. In the future, we plan to more closely examine the accuracy of inverse planning for knowledge diagnosis; this study provides some evidence of its effectiveness, but a more thorough investigation is needed. Additionally, we plan to explore the use of inverse planning for customizing feedback, and to compare the effectiveness of the immediate and specific *corrected* interface feedback to holistic, conceptual feedback. Investigation of a blend of structured and freeform interfaces may also suggest ways to maintain engagement and motivation while still guiding learners towards normative solving. While many questions remain, the technical innovations of inverse planning provide a foundation for addressing these issues.

Acknowledgements. This work was funded by NSF grant number DRL-1420732 to Thomas L. Griffiths.

References

1. Anthony, L.: Developing handwriting-based Intelligent Tutors to enhance mathematics learning. Ph.D. thesis, Carnegie Mellon University (2008)
2. Anthony, L., Yang, J., Koedinger, K.R.: Evaluation of multimodal input for entering mathematical equations on the computer. In: CHI'05 Extended Abstracts on Human Factors in Computing Systems. pp. 1184–1187. ACM (2005)
3. Baker, C.L., Saxe, R.R., Tenenbaum, J.B.: Action understanding as inverse planning. Cognition 113(3), 329–349 (2009)
4. Board, C.: Accuplacer® sample questions for students. `https://accuplacer.collegeboard.org/students`, accessed: 2014-10-15
5. Corbett, A.T., Anderson, J.R.: The effect of feedback control on learning to program with the lisp tutor. In: Proceedings of the Twelfth Annual Conference of the Cognitive Science Society. pp. 796–803 (1990)
6. Falmagne, J.C., Doignon, J.P.: Learning spaces. Springer (2011)
7. Gilks, W., Richardson, S., Spiegelhalter, D.J. (eds.): Markov Chain Monte Carlo in Practice. Chapman and Hall, Suffolk, UK (1996)
8. Koedinger, K.R., Anderson, J., Hadley, W., Mark, M.: Intelligent tutoring goes to school in the big city. International Journal of Artificial Intelligence in Education 8, 30–43 (1997)
9. Kramer, A.: Symja library - Java symbolic math system. `https://bitbucket.org/axelclk/symja_android_library/wiki/Home` (2010-2014)
10. Lund, A.: Measuring usability with the USE questionnaire. STC Usability SIG Newsletter 8(2) (2001)
11. Mattern, K.D., Packman, S.: Predictive validity of ACCUPLACER scores for course placement: A meta-analysis. Tech. Rep. 2009-2, College Board, New York, NY (December 2009)
12. Payne, S., Squibb, H.: Algebra mal-rules and cognitive accounts of error. Cognitive Science 14(3), 445–481 (1990)
13. Rafferty, A.N., Griffiths, T.L.: Diagnosing algebra understanding via Bayesian inverse planning. In: Proceedings of the 7th International Conference on Educational Data Mining. pp. 351–352 (2014)
14. Rafferty, A.N., LaMar, M.M., Griffiths, T.L.: Inferring learners' knowledge from their actions. Cognitive Science (in press)
15. Razzaq, L.M., Heffernan, N., Feng, M., Pardos, Z.: Developing fine-grained transfer models in the Assistment system. Journal of Technology, Instruction, Cognition, and Learning 5(3), 289–304 (2007)
16. Sleeman, D.: An attempt to understand students' understanding of basic algebra. Cognitive Science 8(4), 387–412 (1984)
17. Sutton, R.S., Barto, A.G.: Reinforcement learning. MIT Press (1998)