An ACT-R/PM Model of Algebra Symbolization

Kevin A. Gluck
Air Force Research Laboratory
Warfighter Training Research Division
Mesa, AZ 85044 USA
(kevin.gluck@williams.af.mil)

Abstract

This paper reports empirical evidence corroborating the presence of an *inductive support effect* (Koedinger & Anderson, 1998) in algebra symbolization. Students who solved one or more result-unknown problems before symbolizing were faster and more accurate symbolizers than students who completed the symbolization first. Eye movement data show that the primary process difference is in reading the problem statement. The second half of the paper describes an ACT-R/PM (Byrne & Anderson, 1998) model of correct symbolization in the inductive support condition. Although it represents only a limited subset of the data, the model does capture the average completion time and fixation frequency data, and it serves as a starting point for testing alternative process models that interact directly with the tutoring software.

Introduction

Algebra symbolization is the translation of a word problem into an algebraic expression that represents the quantities and relationships among quantities in the problem. For instance, let's say a student were presented with the following problem situation:

> "Concert tickets cost 45 dollars a piece. A friend offers to stand in line for a number of tickets, if you will pay him a fee of 12 dollars to do so."

If the student wrote a symbolic expression for that problem situation (e.g., 20-4x), she would be *symbolizing*. Symbolization is a skill that is useful in a variety of modern tasks, like creating spreadsheets and computer programming. Symbolization also is a skill that has been receiving a good deal of empirical attention recently (Gluck, 1999; Heffernan & Koedinger, 1997, 1998; Koedinger & Anderson, 1998).

Koedinger and Anderson (1998) reported a performance advantage in symbolization when it followed the completion of result-unknown questions. *Result-unknown* questions are those that provide a value (the *given*) for the changing quantity (*hours* in the problem above) and require that the student arrive at the resulting solution. An example of a result-unknown question based on the problem statement above might be: "What will be the total cost for 3 tickets?" Koedinger and Anderson found that students who symbolized after solving two result-unknown problems were able to symbolize faster during tutoring and also showed better overall learning gain from pretest to posttest than students who symbolized before solving the result-unknown problems. They refer to these positive findings as resulting from the *inductive support effect*. Their interpretation is that students are able to induce the appropriate algebraic symbolization of the problem statement out of the arithmetic operators used in solving the result-unknown problems, and this leads to better performance and better learning.

In recently-completed dissertation research (Gluck, 1999), I attempted to replicate these results. Students were assigned to either the Inductive Support or the Non-Inductive Support condition in a between-subjects manipulation. This research happened in the context of an investigation into the instructional opportunities that are available by combining eye tracking and intelligent tutoring, so eye movement data were collected in addition to error and latency data. This paper describes both the results related to the Inductive Support effect and also an ACT-R/PM model of algebra symbolization in the Inductive Support condition.

Method

Participants

A total of 18 middle-school and high-school students completed this study. Half of the participants were female and half were male. The students ranged in age from 12 to 15 years old (6^{th} to 9^{th} grade). All of the participants had progressed at least as far as Pre-Algebra in their mathematics studies, and 13 were taking or had completed an Algebra class. One student was currently enrolled in an Algebra 2 class. All participants were paid $50 for completing the study.

The EPAL Algebra Tutor

EPAL stands for Eye Point-of-regard Analysis Laboratory. The EPAL Algebra Tutor is a streamlined recreation of the Worksheet tool, as it existed in the Practical Algebra Tutor (PAT) used in the Koedinger and Anderson (1998) study. The Worksheet is a tabular spreadsheet interface that holds a problem statement and questions. The student's task is to fill in the column labels and units, enter a variable and an expression written in terms of that variable, and then answer two questions. There were 16 different EPAL Algebra Tutor problems. Completion of the problems proceeded from top to bottom, which was important for testing for an effect of inductive support. With respect to that manipulation, the important thing to know is that in the Non-Inductive Support condition, students did the symbolization *before* answering result-unknown questions, and in the Inductive Support condition, students did the symbolization *after* answering result-unknown questions.

Design and Procedure

Upon arrival at the laboratory on Day 1, there was a brief eye-tracker calibration test, to confirm that we would be able to acquire a reasonably accurate and stable image of the student's eye. With this confirmed, the student completed a short demographic survey, then a paper-and-pencil pretest. The final activity on Day 1 was an introduction to the EPAL Algebra Tutor, by way of an introductory problem, to familiarize the student with the interface and the type of problems to be encountered on subsequent days. Day 2 started with the completion of a second introductory problem, then there were four tutor problems while calibrated on the eye tracker. These were randomly selected without replacement from the pool of 16 problems. Days 3 and 4 simply involved the completion of four more problems each day. Day 5 started with the completion of four problems (the last 4 in the set of 16), followed by the posttest and debriefing.

Results

Completion Time

Koedinger and Anderson (1998) reported an effect of inductive support on symbolization completion time during learning. On two-operator problems similar to those in the EPAL Algebra Tutor, students in their Inductive Support condition were considerably faster ($\underline{M}$ = 28 s) at symbolizing than were students in the Textbook (Non-IS) condition ($\underline{M}$ = 48 s). They found no effects of inductive support on problem solving time in either result-unknown or start-unknown problems.

This effect is replicated in the EPAL Tutor data, with symbolization time faster in the IS condition ($\underline{M}$ = 14.8; $\underline{SD}$ = 4.2) than in the Non-IS condition ($\underline{M}$ = 25.1; $\underline{SD}$ = 10.9): $F(1, 16) = 6.95$, $p < .02$. The effect is still significant after controlling for individual differences in pretest performance. This provides corroborating evidence that during learning, students in the Inductive Support condition are faster at symbolizing than are students in the Non-Inductive Support condition.

Errors

Koedinger and Anderson (1998) did not report an effect of the inductive support manipulation on error rates. In the EPAL Tutor data, I found that symbolization error rates were lower in the IS condition ($\underline{M}$ = .10; $\underline{SD}$ = .20) than in the Non-IS condition ($\underline{M}$ = .41; $\underline{SD}$ = .14). This is consistent with the inductive support effect, and is a statistically significant performance advantage: $F(1, 16) = 14.90$, $p < .01$. The result holds even when controlling for pretest differences.

Latency Differences During Correct 1st Attempts at Symbolizing

Students in the Inductive Support condition show a lower error rate on expression symbolization, which could entirely account for their faster latencies. Is there any latency advantage for the Inductive Support condition in the absence of errors?

This question has been addressed by looking at data just from correct 1st attempts at symbolizing. An average correct 1st attempt time was computed for each student, and these 18 data points serve as the basis for this analysis. The mean latency for students in the high formula row condition (Non-Inductive Support) was 21.24 seconds, whereas the mean latency for students in the Inductive Support condition was 12.81 seconds. This difference is statistically significant: $F(1, 16) = 9.80$, $p < .01$, and holds up when the individual differences on the pretest are covaried out. So it seems that even in the absence of errors, there is a latency advantage for the Inductive Support condition.

Fixation Analyses

The data clearly suggest a performance advantage during learning for students in the Inductive Support condition. The completion time data from correct 1st attempts are especially convincing with respect to that conclusion. Another revealing source of data are students' eye movements. The eye movement data can be used to address the question of what it is that is giving the Inductive Support students this latency advantage. What are they

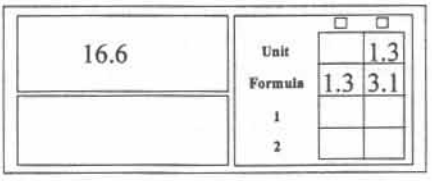
doing differently? What do the eye movement data tell us about how the solution process differs depending on the presence of inductive support?

Fixation data were extracted from correct 1st attempt part-tasks in the expression cell, separately by condition. This provided a fixation frequency for every POR region (separately for high and low formula row conditions) and these numbers were divided by the total number of part-tasks to arrive at an average fixation frequency per part-task. There were 87 part-tasks from the Non-Inductive Support condition and 124 part-tasks from the Inductive Support condition, reflecting the fact that students in the IS condition generally found it easier to symbolize. Figure 1 is a generic representation of the worksheet as it appears in the two conditions. Cells that averaged more than 1.0 fixation per part-task are labeled with their fixation counts.

In Figure 1a, there is a cluster of fixations around the cell Formula-Right, which is the cell under edit. This is where the expression is being entered into the spreadsheet. Most importantly, however, note the average of 16.6 fixations in the problem statement. This reflects a good deal of reading, and in fact reflects *re-reading* the problem statement, since students had to first read it to enter the column labels and units.

Figure 1b is the Inductive Support condition. The most striking difference in the fixation frequencies is in the reading of the problem statement. Fixations in the problem statement drop to 3.0 per part-task in the IS condition, meaning that students spend considerably less time reading the problem statement. The only other POR region to receive more than 1.0 fixation per part-task is Formula-Right, the cell under edit.

1a. No Inductive Support



1b. Inductive Support

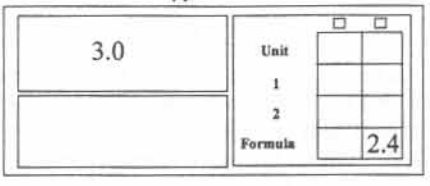


Figure 1 Mean fixation counts greater than 1 during symbolization in the Non-IS condition.

Modeling Symbolization in the Inductive Support Condition

The cognitive analysis that motivated the Koedinger and Anderson (1998) work was focused at the strategic level (e.g., arithmetic translation strategy, inductive-support strategy, algebra translation strategy) and not at the production rule level. Ohlsson (1998) correctly notes that they do not provide a cognitive model of the symbolization process

itself, and this project provides an opportunity to move in the direction of a running computer model of symbolization.

The model to be presented here is a performance model of symbolization in the Inductive Support condition. It makes especially good sense to model performance in the Inductive Support condition because the results from this study and the Koedinger and Anderson (1998) work suggest that an inductive support design is a better way to teach symbolization. Given this, it seems the odds are increasingly likely that future versions of the Algebra tutor will be designed this way, and the production rules used to model symbolization should reflect that design.

ACT-R/PM

Although a brief summary of the architecture is appropriate here, Byrne and Anderson (1998) should be consulted for more information on the ACT-R/PM architecture. ACT-R/PM is designed as a group of modules that control cognition, perception, and motor movements. The cognitive module is ACT-R 4.0 (Anderson & Lebiere, 1998), and the remaining modules are a re-implementation of the analogous modules in EPIC (Meyer & Kieras, 1997).

The perceptual modules, vision and audition, take input from the environment (i.e., computer) and make it available in the form of either an icon (vision module) or an audicon (audition module). Only the visual icon is relevant in this symbolization model. Production rules move attention around the visual icon and attend to its contents. This results in the creation of declarative chunks representing the objects and locations of the objects that make up the contents of the visual field.

Attentional shifts and all motor and speech acts are directed by production firings. A production that sends a MOVE ATTENTION command results in the updating of a chunk in declarative memory, whereas a production that calls for a motor or speech act results in an action that can change the state of the environment (e.g., a mouse click or a key press). If the environment (computer screen) does change, the change is not reflected in declarative memory until attention is moved to the portion of the environment that changed and the new state is encoded.

A Model

The model begins at the point where a student has just entered the 'x' as the variable for the left-hand column, and is about to select the expression cell. So to be still more accurate about exactly what is being modeled here, this is a model of 1st attempt symbolization among students in the Inductive Support condition. The model is limited to 1st attempts currently because it does not engage in an error recovery process after it makes a mistake. Since the model represents a student in the Inductive Support condition, each time the model is initialized the problem sets up with the result-unknown cell(s) and the cells for the column labels and units of measure already completed.

Influences and Assumptions in Designing the Model

Heffernan and Koedinger (1997, 1998) have described symbolization as analogous to translation. This translation is from the familiar language of the problem statement to the foreign language of algebra. Translation is a two-step process, involving *comprehension*

and *production*. Students must first comprehend the problem statement, and then take the representation they have formed and use it to produce an expression.

Because this is a model of students in the Inductive Support condition, the assumption is made that there already exists in declarative memory an accurate representation of the problem quantities and relations in the problem statement. Conceptually, this representation can be thought of as a directed quantitative network like that described by Tabachneck, Koedinger, and Nathan (1995). Tabachneck et al. propose a *directed* quantitative network in which "quantities are represented as nodes and constraints as 3-part directed relations where the quantity at the arrow is the *output* and the other two quantities are *inputs* that are combined with the arithmetic operation to produce the output" (p. 398). It is assumed that a representation of the problem statement that is analogous to a directed quantitative network exists in the student's knowledge base by the time symbolization is required in the Inductive Support condition.

Chunks

This sort of a network representation is fine at a conceptual level, but at the implementation level some additional decisions have to be made regarding how to represent the network. It is assumed in the model that this network of quantities and relations in the Tickets problem is stored declaratively. Declarative knowledge in ACT-R is represented in chunks. A *chunk* is a single unit of declarative knowledge, and all chunks have their origins either in perceptual encodings of the environment or in the encoding of past goals. One way to think about chunks is that they represent a pattern of information in a person's declarative knowledge base, and this makes a chunk representation a particularly appealing one for the pattern of information in the directed quantitative network.

The particular chunk representation used in this model was strongly influenced by the problem representations that are generated by MacLaren and Koedinger's (1996) Early Algebra Problem Solving 2 (EAPS2) model. These chunks provide a linked list structure that can be used to recall the appropriate values and operators when it is time to symbolize. By itself, however, declarative knowledge in ACT-R is inert. It needs productions in order to be of any real service to cognition, and in fact productions are necessary in order for cognition to take place at all.

Productions

In this model, the locus of control over the order in which the components of the expression (intercept, operator, slope, x) are typed is in the goal chunks. Through modifications of the goal chunks, the model dictates which next action will take place. When it needs to type something, the model sets a subgoal, executes the appropriate motor commands, and pops the subgoal to move on to the next component of the expression. Currently, the model is totally deterministic. It always produces the same correct expression, and it always uses the same procedure.

Process

As mentioned earlier, the model initiates in a state where the variable cell (where the 'x' is typed) has just been completed. The model's hand is now on the mouse (in anticipation of selecting the next cell). Students typically execute "confirmation" saccades

to the cell that has just been completed before selecting the next cell, and the first thing the model does is move visual attention to the 'x' and confirm that it is there. The model then moves visual attention to the expression cell, moves the mouse to the expression cell, and selects the cell with a mouse click. Now the model is ready to begin creating the expression. It retrieves the intercept out of declarative memory and moves visual attention to the problem statement to confirm that the retrieved value actually did occur in the problem. With the confirmation complete, the model looks at the keyboard and types the intercept. The model then retrieves and types the operator representing the sign of the slope (+ or -). It retrieves the value for the slope, looks at the problem statement to confirm that value, shifts visual attention to the keyboard again, and types the slope. The last step in creating the expression is to type the variable, which the model does, then it moves visual attention back to the expression cell and hits the return key to enter the answer.

Comparison of Model and Student Performance

In the Inductive Support condition, which is the condition the model is presumed to be in, 88% of the 1st attempts at symbolization were correct responses. Since the bulk of the data are correct responses, the performance comparison will focus on the characteristics of the correct performance model in comparison with the correct student part-tasks.

Table 1 shows the average symbolization completion time data for all of the correct student part-tasks and 10 runs of the model.

Table 1
Average Symbolization Completion Time Data for Students and Model

	Completion Time (seconds)		
	M	(SD)	n
Students	13.2	(7.8)	124
Model	13.1	(.2)	10

As is clear in Table 1, the model is right on with respect to matching the mean completion time, but there is considerably less variability in the model's performance times than there is in the students. This is consistent with the fact that there is considerably less variability in the model's completion process than there is in the students.

Regarding fixation frequencies, the number of fixations the model makes to both the expression cell and the problem statement are in line with the average fixation frequencies seen in the students. The model looks to the expression cell 2 times (compared to the students' 2.4 fixations) and it looks to the problem statement twice (compared to the students' 3 fixations).

The two fixations in the expression cell occur when attention first moves to that cell to initiate symbolization and when the model looks back to the cell again after the expression is entered (just before hitting the return key). These fixations are labeled "Look at Exp. Cell" in Figure 3. The fixations in the problem statement come from confirmation looks to the intercept and slope values. The model is designed to look for those values to confirm that it retrieved the correct values before typing them in.

I do not intend to suggest by this model that all students symbolize in exactly this way, but the model as it currently is designed does provide a good match to both the latency and the fixation data from this sample of students. Clearly it is the case that the model could be expanded to provide opportunities for alternative solution procedures, and this is an obvious direction for future modeling work in this domain. The current model provides a solid base from which to begin exploring additional and alternative model designs.

Conclusion

This paper has provided a running model of algebra symbolization. The model uses the default cognitive and perceptual-motor parameter settings in ACT-R/PM, and these provide a good match to the completion times displayed by successful students in this condition. This model assumes that the effect of inductive support is to create an accurate and retrievable declarative representation of the mathematical structure of the problem. Thus, the subject no longer has to search the problem to create this representation but only looks to the problem to confirm the relationships already stored in memory.

Acknowledgements

Sincere thanks to John Anderson and Scott Douglass for their contributions to this work. Funding for this project has been provided by the Air Force PALACE Knight Program and by NSF grant number CDA-9720359 to the Center for Interdisciplinary Research on Constructive Learning Environments (CIRCLE)..

References

Anderson, J. R., & Lebiere, C. (1998). The atomic components of thought. Hillsdale, NJ: Erlbaum.

Byrne, M. D., & Anderson, J. R. (1998). Perception and action. In J. R. Anderson & C. Lebiere, The atomic components of thought (pp. 167-200). Mahwah, NJ: Erlbaum.

Gluck, K. A. (1999). Eye movements and algebra tutoring. Unpublished doctoral dissertation, Carnegie Mellon University, Pittsburgh, PA.

Heffernan, N. & Koedinger, K. R. (1997). The composition effect in symbolizing: The role of symbol production vs. text comprehension. In Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society, (pp. 307-312). Hillsdale, NJ: Erlbaum.

Heffernan, N. & Koedinger, K. R. (1998). A developmental model for algebra symbolization: The results of a difficulty factors assessment. In Proceedings of the Twentieth Annual Conference of the Cognitive Science Society, (pp. 484-489). Hillsdale, NJ: Erlbaum.

Koedinger, K. R., & Anderson, J. R. (1998). Illustrating principled design: The early evolution of a cognitive tutor for algebra symbolization. Interactive Learning Environments, 5, 161-179.

MacLaren, B. A., & Koedinger, K. R. (1996) Toward a dynamic model of early algebra acquisition, In the Proceedings of the European Conference on Artificial Intelligence in Education (pp. 38-44). Lisbon, Portugal: Edicoes Colibri

Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. Psychological Review, 104 (1), 3-65.

Ohlsson, S. (1998). Representation and process in learning environments for mathematics: A commentary on three systems. Interactive Learning Environments, 5, 205-215.

Tabachneck, H. J. M., Koedinger, K. R., & Nathan, M. J. (1995). A cognitive analysis of the task demands of early algebra. In Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Erlbaum.