Developing a Pedagogical Domain Theory of Early Algebra Problem Solving

Kenneth R. Koedinger Benjamin A. MacLaren

March 2002
CMU-CS-02-119
CMU-HCII-02-100

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We describe a theory of quantitative representations and processes that makes novel predictions about student problem-solving and learning during the transition from arithmetic to algebraic competence or "early algebra". Our Early Algebra Problem Solving (EAPS) theory comes in the form of a cognitive model within the ACT-R cognitive architecture. As a "pedagogical domain theory", our EAPS theory can be used to make sense of the pattern of difficulties and successes students experience in early algebra problem solving and learning. In particular, the theory provides an explanation for the surprising result that algebra students perform *better* on certain word problems than on equivalent equations (Koedinger & Nathan, 2000; Nathan & Koedinger, 2000). It also makes explicit the knowledge and knowledge selection processes behind student strategies and errors and provides a theoretical tool for psychologists and mathematics educators to both *productively generate* and *accurately evaluate* hypotheses about early algebra learning and instruction. We have abstracted our development process into six model-building constraints that may be appropriate for creating cognitive models of problem solving in other domains.

This research was supported by a grant from the James S. McDonnell Foundation program in Cognitive Studies for Educational Practice, grant #95-11.

Keywords: learning, cognitive modeling, cognitive architecture, problem solving, mathematics education, model-data fit

1. INTRODUCTION

General cognitive architectures, like ACT-R (Anderson and Lebiere, 1998), Soar (Newell, 1990), neural network models (O'Reilly and Munakata, 2000), etc., can be important aids to cognitive engineering. However, given the variety of different kinds of knowledge and strategies that are possible within such cognitive architectures, a knowledge-specific layer of theory development is also necessary, particularly for complex problem solving domains like those that are the subject of instructional design efforts. While cognitive architectures constrain the development of such theories, knowledge representation choices must be made that are under-constrained by the architecture, but are critical to the explanations and predictions generated from the theory.

The degrees of freedom in knowledge representation choices have less impact on applying cognitive architectures in simpler domains employing more routine cognitive skills (e.g., the text-editing models in Card, Moran, & Newell, 1983). The need for additional knowledge-specific theoretical generalizations, beyond the knowledge-neutral generalizations provided by the architecture, increases with the cognitive complexity of the domain of concern. Knowledge-specific generalizations are particularly critical when applications, like instructional design, depend on modeling multiple strategies and variations in performance and errors due to learning.

We use the term "pedagogical domain theory" to refer to a theory that is constrained by cognitive architecture but, more importantly, further captures relevant generalizations about the nature of knowledge and learning in a domain of interest. This paper describes a pedagogical domain theory for "early algebra problem solving" and illustrates specific ways in which the ACT-R architecture has constrained this theory development. This theory specifies knowledge-specific generalizations about quantitative representations and processes that yield explanations of and predictions about student problem-solving and learning behavior during the transition from arithmetic to algebraic competence. This transition phase is referred to as "early algebra" (Kaput, 1999). Our theory of early algebra problem solving (EAPS) comes in the form of a computer simulation based on the ACT-R theory and written within the ACT-R cognitive modeling software (Anderson and Lebiere, 1998). We wanted a theory that not only explains and predicts the *kinds of steps* students generate in successful problem solving, but also explains and predicts *incorrect* steps and the *frequency* with which different kinds of correct and incorrect steps are generated.

We describe the approach we used to combine empirical data, mathematical modeling, and cognitive task analysis to create our theory and cognitive model. We have abstracted our theory development process into six model building constraints that may be appropriate to building cognitive models of problem solving in other domains.

From an applied research viewpoint, we wanted a theory that would make sense of the pattern of difficulties and successes students experience in early algebra problem solving and learning. In particular, we wanted an explanation for the surprising result that algebra students perform *better* on certain word problems than on equivalent equations (Koedinger & Nathan, 2000; Nathan & Koedinger, 2000). Our theory provides a detailed cognitive process explanation for this result that makes explicit the knowledge and knowledge selection processes behind student strategies and errors. It also provides a powerful theoretical tool for psychologists and mathematics educators to both *productively generate* and *accurately evaluate* hypotheses about early algebra learning and instruction.

From a basic research viewpoint, the development of this theory provides a test of the adequacy of the ACT-R theory and an indication of what features of ACT-R can be used to model complex problem solving choices in a realistic task domain. These critical features may (or may not) be shared by other unified theories of cognition and thus provide a basis for comparison.

The development of theories of domain knowledge, particularly for core knowledge domains like quantitative reasoning, should be within the purview of basic research within cognitive science. For any domain of real consequence, predictions of problem-solving performance and learning are impossible without a rich theory of knowledge within that domain.

1.1. Constraints of a Cognitive Model of Problem Solving and Learning

There are two traditional methods for creating cognitive models: task analysis and verbal protocols (Newell & Simon, 1972). Task analysis is a highly subjective process, while verbal protocols rely on detailed analysis of lengthy transcriptions of problem-solving episodes. The approach we took relies less on the task analyst's intuition, yet does not require verbalized descriptions of problem-solving behavior, that are often difficult for students to produce and difficult for researchers to interpret. Our theory development progressed through a set of constraints we wanted the final model to satisfy.

We abstracted six constraints from the literature and from our experience in developing our theory of early algebra problem solving (EAPS) shown in Table 1.

Table 1. Constraints on Cognitive Model Development Relevant to Creating Pedagogical Domain Theories.

C1. **Solution Sufficiency**. At minimum, a cognitive model of problem solving should be able to successfully solve problems in the domain being modeled.

C2. **Step Sufficiency**. A cognitive model should compute steps in behavior that qualitatively match the kinds of correct and incorrect steps in the solution traces of human problem solvers (Newell and Simon, 1972; Anderson, 1993). [1]

C3. **Choice Matching**. A model should be able to capture the statistical structure of subject choice behavior, that is, a model should account for the frequency of alternative decisions, solution strategies, or errors subjects may make. This constraint reflects a measure of goodness of fit (Polk, VanLehn, & Kalp, 1995; Ritter, 1992; Ritter and Larkin, 1994; Salvucci, 1999).

C4. **Computational Parsimony**. Explicit representation of skills that have no computational role and no qualitative or quantitative predictive value should be eliminated. In statistical terms, we prefer models that minimize the number of parameters needed.

C5. **Acquirability**. A model should not be considered complete if it only captures data about student performance. It must also reflect how students acquire knowledge in a domain,. Components in the model should be modified if it is not possible to tell a plausible story about how those components might have been learned (c.f., Newell, 1990, p. 308; Anderson & Lebiere, 1998, p. 109, 343).

C6. **Transfer**. The model should provide an account for how knowledge acquired in one context can apply to other contexts where it is relevant. Components of the model should be written at a small enough grain size so that similarities in computation across different problem-solving contexts are captured by overlap in the use of specific knowledge components. These similarities should not be hidden within aspects of larger encapsulated knowledge components (Singley and Anderson, 1989).

In this article we use these constraints as an organizing framework for describing the history of the development of EAPS. Before describing this development, the remainder of this section provides relevant theoretical and empirical background. Then, in section 2, we describe our initial EAPS model and how it met the Solution and Step Sufficiency Constraints (C1 and C2). In section 3, we describe how we adapted the model to meet the Choice Matching and Computational Parsimony Constraints (C3 and C4). In section 4, we describe further modifications to meet the Acquirability and Transfer Constraints (C5 and C6). At each stage different empirical and analytic approaches were brought to bear. These approaches include qualitative matching against student problem-solving behavior (to meet C1, C2 and C6), quantitative data fitting (C3), and analytic inspection of computational characterizations of the production rules (C4, C5, C6). Section 5 provides a general discussion and section 6 concludes the paper.

[1] Verbal protocols add information beyond solution traces when students verbalize intermediate mental steps like subgoal, so the extent to which this step sufficiency constraint reflects students' internal mental structures depends on the level of analysis being done.

1.2 Modeling Complex Problem-Solving Choices in ACT-R

Our modeling work is implemented in ACT-R, a rich and flexible cognitive architecture (Anderson, 1993). ACT-R breaks knowledge into two main categories, a declarative knowledge base of facts, and a procedural knowledge base of production rules. A production rule is a simple condition-action or IF-THEN statement that is used for creating new declarative facts or modifying existing ones. Productions have the following general form: "If there is a goal G and some context C, then perform action X or set goal Y."

One central issue in production-based cognitive architectures like ACT-R is how to deal with multiple productions that are simultaneously applicable to the same state of the world. If the condition side of several productions applies, there is a conflict that needs to be resolved. To model these situations, ACT-R includes a "rational" component for conflict resolution based on decision theory. To determine which production to fire if there is a conflict, the expected gain of each production is computed. The expected gain or utility of a production is defined as: Expected Gain = PG–C, where G is the estimated value of the current goal, P is the estimated likelihood that executing the production will eventually satisfy the current goal and C is the estimated cost (or cognitive effort) of executing the production.[2] In ACT-R, the production with the highest expected gain is not always chosen. Rather there is a stochastic process, implemented as a Gaussian noise parameter, that will sometimes cause a production to be selected other than the one with the highest estimated utility. Production rule utilities provide a useful approach to modeling student choice behavior.

1.3 Review of Student Data: A Difficulty Factor Assessment of Problem-solving

As part of a broader research effort to provide a scientific basis for improved mathematics instruction (e.g., Koedinger & Anderson, 1993; Koedinger, Anderson, Hadley, & Mark, 1995), we have been performing detailed empirical and theoretical investigations of students' developing quantitative problem-solving skills. We have been performing experimental studies, called "difficulty factors assessments" (Koedinger & Tabachneck, 1995), and are using the ACT-R theory and software (Anderson, 1993) to create detailed models of algebraic competence and its development.

Our empirical studies of early algebra have established a striking contrast between students' difficulties with symbolic algebra and their relative success with certain kinds of "intuitive" algebraic reasoning. Much to the surprise of most math teachers and educators (Nathan & Koedinger, 2000), high school algebra students in our studies are better able to solve certain algebra word problems than the corresponding algebra equation.

A "Difficulty Factor Assessment" (DFA) involves the systematic investigation of problem factors that may lead to student difficulties in problem solving. The ACT-R models we report here attempt to account for the effects of three factors in data from two DFA studies (Koedinger & Nathan, 2000). Two of these factors are illustrated in Figure 1, unknown position and presentation type. The pairs of problems in each row of Figure 1 differ in where the problem unknown is positioned. The problems in column 1 are called "result-unknown" problems because the unknown is the result of the process described. The problems in column 2 are "start-unknown" problems because the unknown is the start of the process described. Problems within the columns illustrate a second factor. They require the same underlying arithmetic, but differ in the manner in which they are presented. The "Story Problems" in the first row are presented verbally and include reference to a real world situation (e.g., wages). The "Word Equations" in the second row are also presented verbally but do not include a situation. The "Equations" in the third row are presented symbolically and have no situational information. Other factors we have looked at that are not illustrated in Figure 1 include number difficulty (integers versus decimals) and the cover story used in different story problems (e.g., the "waiter" story in Figure 1).

---

[2] ACT-R does not propose that people explicitly analyze their problem-solving choices using decision theory explicitly, but rather that their choices conform with this analysis.

	Result-Unknown Problems	Start-Unknown Problems
Story Problems	When Ted got home from his waiter job, he multiplied his hourly wage, \$2.65, by the 6 hours he worked that day and added the \$66 he received in tips. How much money did Ted make that day?	When Ted got home from his waiter job, he took the amount he made that day and subtracted the \$66 he made in tips. He divided the resulting amount by the six hours he worked and got \$2.65, his hourly wage. How much did Ted make that day?
Word Equations	If I multiply 2.65 by 6 and then add 66, I get a number. What number do I get?	Starting with some number, if I subtract 66 and then divide by 6, I get 2.65. What number did I start with?
Equations	2.65 * 6 + 66 = X	(X – 66) / 6 = 2.65

Fig. 1. Example Combinations of Difficulty Factors

Two DFA studies of early algebra problem solving (DFA1 and DFA2) were performed with students near the end of a yearlong high school algebra class. The studies revealed large effects for unknown position, problem presentation and number difficulty (integers vs. decimals). Not surprisingly students are significantly better at result-unknown (arithmetic) problems than start-unknown (algebra) problems and are significantly better at problems with integer quantity values than problems with decimal quantity values. However, it comes as a surprise to many that these algebra students had the greatest difficulty with the equations which were significantly harder than the word equations ($p<.001$ in both studies) which in turn were only slightly harder than the story problems ($p=.23$ in DFA1 and $p<.01$ in DFA2). The effects of these three factors are for the most part independent and additive.

Prior models of algebra story problem solving (e.g., Bobrow 1968; Lewis 1981) have employed a two-step process. Story problems are converted into equations and the equations are then solved using symbolic algebra. Figure 2 shows a student from DFA1 using equation solving. The student writes an equation ($25x + 10 = 1.10$) and then explicitly manipulates it, subtracting ten from both sides, and then dividing both sides by 25 to arrive at an answer. Such a model predicts that performance on story problems must be worse than performance on equations since equation solving is a subgoal of story problem solving. Most teachers and math educators share this same prediction (Nathan & Koedinger, 2000). Students' greater success on word problems in DFA1 and DFA2 indicates they must not be following this two step process. Our results showed that they are often using alternative informal methods for finding answers.

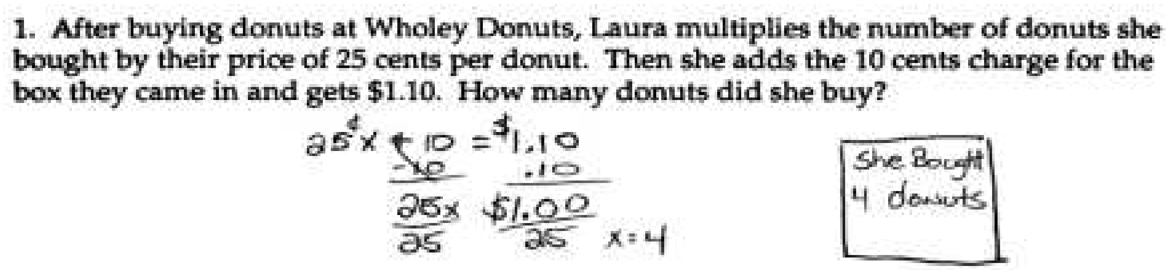

Fig. 2. Formal algebra on Story Start-unknown Problem

Students translated verbal problems to algebra equations on only 13% of problems and 53% of these attempts led to success. More often students attempted to solve verbal start-unknowns using one of two informal strategies, "guess-and-test" or "unwind." In guess-and-test, a value for the unknown is guessed at and that value is propagated through the known quantitative relationships. The guess is then adjusted and the process repeated until the correct answer is arrived at. Guess and test was used about 20% of the time on the verbal start-unknown problems.

By far the most common strategy, informal "unwind", was used almost 40% of the time. Unwind is a verbally mediated strategy, where students work backwards from the given result value, inverting operators along the way, to produce the unknown start value. Figure 3 shows an example of the unwind strategy being used on a decimal story start-unknown problem. The student skips past the algebraic formulation and directly performs the required arithmetic operations. The operations used in the solution are the inverse of those described in the problem and are performed one operator at a time in the reverse order of the problem description.

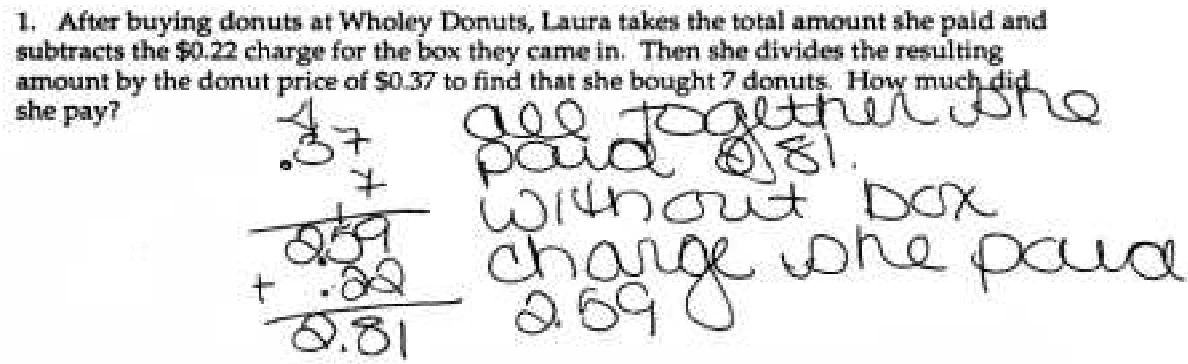

Fig. 3. Unwind on a Decimal Start-unknown Problem

2. EAPS1: MEETING THE SOLUTION SUFFICENCY (C1) AND STEP SUFFICENCY (C2) CONSTRAINTS

2.1 Overview of the Early Algebra Problem Solving Models

We have created a series of "Early Algebra Problem Solving" (EAPS) models of these data that have involved both empirically and theoretically driven refinement. In developing the first EAPS model, EAPS1, our major goal was to satisfy constraints C1 and C2, Solution Sufficiency and Step Sufficiency. In EAPS2 we focused on choice matching, on getting the model to make accurate quantitative predictions about the frequency of student strategies and errors.

We chose ACT-R to address these goals because production rules provide a way to decompose the problem-solving process and predict individual problem steps observed in student solution traces. ACT-R also provides a way to model how productions are selected and how selection between competing productions leads to different strategies and errors.

One assumption in all of our modeling work is that a student represents the problem internally by a set of quantitative relations between the quantities in the problem. The approach builds on prior cognitive analyses of mathematical problem solving (Hall, Kibler, Wenger & Truxaw, 1989; Shalin & Bee, 1985; Mayer, 1982; Paige and Simon, 1966). Each quantitative relation has a pair of input quantities, a mathematical operation, and an output quantity representing the result of that operation applied to the inputs. The output or any of the inputs can be unknown. Example quantitative relation networks are shown in Figure 4. Quantities are displayed in boxes and quantitative relations are displayed by arrows connecting them with a mathematical operation. On the left we show the quantitative network for a result-unknown story problem from Figure 1, where all of the inputs are known. On the right is the quantitative network for the analogous problem in equation form. Notice that even though the problems have the same underlying structure, the story problem on the left has more verbal and situational information associated with it. For example, quantities in the story representation on the left have situational labels (e.g., tips) and units (e.g., dollars).

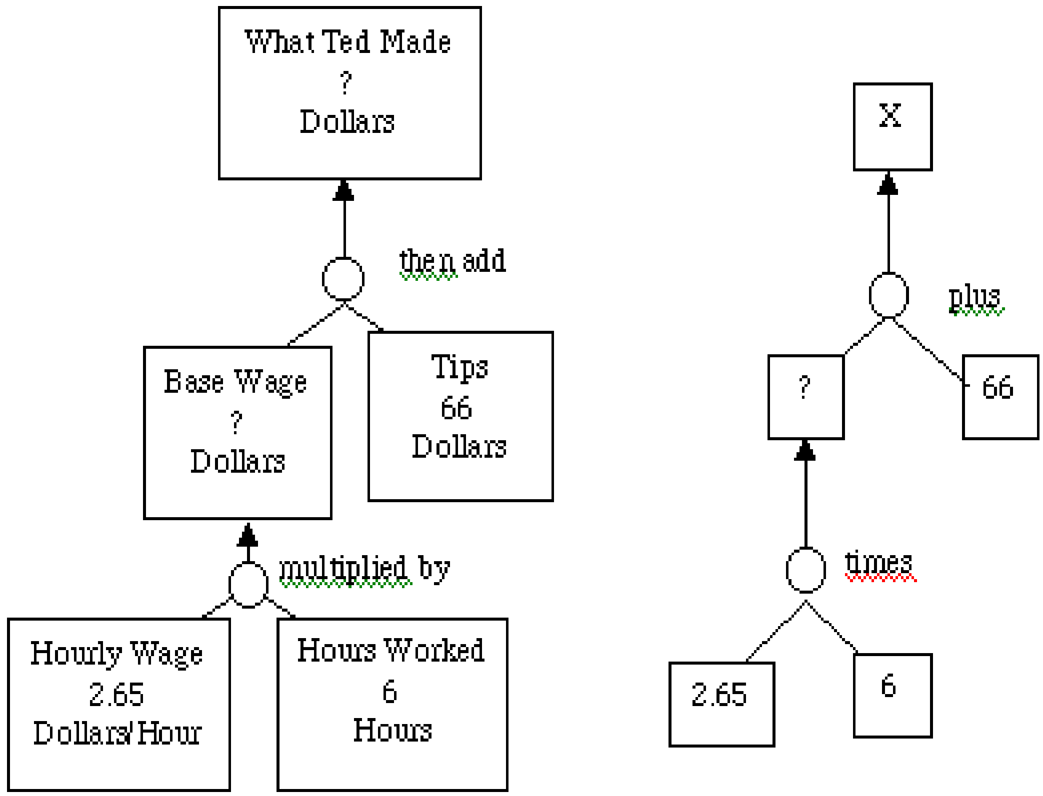

Fig. 4. Quantitative networks for story and equation result-unknown problems in Fig 1.

In the ACT-R model, the network is made up of two kinds of "working memory elements"[3]: Quantities, the rectangular nodes; and Quantitative Relations, the linking circles. How are these constraint networks created and processed? A basic high level description of the control flow in our EAPS theory is summarized in Figure 5. The first major step (#1) describes the process of comprehending an external representation to create internal representations of quantities and a quantitative relation connecting them. This comprehension process includes identifying the arithmetic operation (add, subtract, multiply, divide) and the inputs to and output of that operation. The model can optionally translate the relation into algebraic symbols (step #1c).

In step #2, if one of the inputs is unknown, EAPS deals with it in one of three ways. It can guess a value for the input, it can invert the operation in the relation using the informal unwind strategy, or, if the representation is in equation form (either because it was translated into an equation or it was presented as an equation) EAPS can solve the relation using formal algebra. Finally, in step #3, once the quantitative relation has been comprehended, and unknown inputs have been dealt with as needed, the model can perform the resulting arithmetic subgoal. The process is repeated for each quantitative relation in a problem.

[3] Samples of the actual ACT-R code that specifies working memory elements and production rules can be requested from the authors.

For each quantitative relation in the problem:

1. Comprehend the external representation and create the next quantitative relation (VC, SC)
   a. Identify the arithmetic operation (add, subtract, multiply, divide)
   b. Identify the inputs to and output of that operation
   c. Optionally, translate relation to algebraic symbols (TVS, PS)
   d. If only the output is unknown, go to step 3
      Else if only one input is unknown go to step 2
      Else try another quantitative relationship and go to step 1
2. Deal with unknown input quantity
   a. Invert the operation (UC)
   b. Solve algebra (depends on having done 1c) (TSS, OB)
   c. Guess a value for the input
3. Perform arithmetic subgoal (AP)

Fig. 5. Top-level description of model: 3 major choice points

2.2. Developing EAPS1

EAPS1 (MacLaren & Koedinger, 1996) was our first implementation of the general approach described above. EAPS1 was made up of about 30 ACT-R production rules. An initial development goal was to create a system that could solve problems like those in Figure 1, in other words, satisfy the Solution Sufficiency Constraint (C1). A second development goal was to have the model solve these problems in steps like those of a student problem solver, in other words, satisfy the Step Sufficiency Constraint.

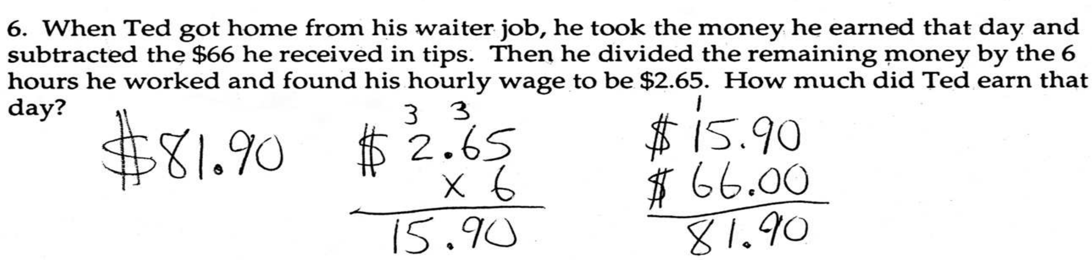

Fig. 6. A decimal arithmetic start-unknown story problem

Figure 6 shows a student employing the unwind strategy to solve a story start-unknown problem. In Table 2 we show a trace of EAPS1 solving this same problem using the same strategy.[4] This trace was produced by "walking" EAPS1 through its space of possible choices in order to replicate the choices reflected in the student solution. Walking stops model execution any time when more than one production can fire, and then presents the different alternatives. The walk routine prompts the user for which alternative should be taken, asking, "What should I do?" and then fires the production corresponding to the user selection.

In step 1 of the trace, EAPS1 is considering which strategy to use or whether to give up. The user guides the model to fire the production SELECT-UNWIND-STRATEGY-ON-VERBAL. At step 2, the production VERBAL-COMPREHENSION-4 is the only one in the conflict set and so it is fired without asking the user. This production identifies the appropriate quantitative relation to focus on and identifies the relevant operator and its inputs and output (corresponding with steps #1a and #1b in Figure 4). At step 3 in Table 2, the model must deal with the unknown input quantity using the selected strategy (corresponding with step #2a in Figure 4). Doing this requires constructing a new goal that can be solved

[4] In our simulations of early algebra problem solving, we did not create a complete model of the process of comprehending the English language input. Instead, at the start of a verbal problem, the simulation's working memory contains the quantitative network that results from the parsing of the English language input.

with straight arithmetic. In the student solution this corresponds to writing the down the arithmetic problem "[$2.65 x 6]" in the middle of Figure 6. Step 4 in Table 2 corresponds with the student solving the resulting arithmetic problem. Steps 5 through 7 are just like steps 2 through 4, comprehending a constraint, inverting the arguments and operator, and performing the resulting arithmetic.

Note that the explicit strategy selection in Step 1 applies to both quantitative relations. Thus, once a strategy was selected in EAPS1 it was pursued throughout the problem. EAPS1's explicit strategy selection and strategy following will be contrasted in section 3.2, with the approach taken in future models.

Table 2. EAPS1 unwinding a verbal problem

```
--- Options ---
1. I could try verbal unwinding.
2. I could try algebra.
3. I'll just give up on the problem.
    What should I do? 1
>>> Step 1: SELECT-UNWIND-STRATEGY-ON-VERBAL
  Let's try unwinding...

>>> Step 2: VERBAL-COMPREHENSION-4
   DIVIDED-BY 6 gives 2.65...

--- Options ---
1. The inverse of DIVIDED-BY is MULTIPLY.       (Unwind Correct)
2. The inverse of DIVIDED-BY is DIVIDED-BY.     (Unwind Error)
    What should I do? 1
>>> Step 3: VERBAL-INVERT-OPERATOR
   I need to take 2.65 and MULTIPLY 6...

--- Options ---
1. 2.65  *  6 = 15.90   (correct with situational support)
2. 2.65  *  6 = 159     (bug)
3. 2.65  *  6 = 0.16    (slip)
    What should I do? 1
>>> Step 4: SITUATED-ARITHMETIC
   2.65  *  6 = 15.90

>>> Step 5: VERBAL-COMPREHENSION-4
   MINUS 66 gives 15.90...

--- Options ---
1. The inverse of MINUS is PLUS.        (Unwind Correct)
2. The inverse of MINUS is MINUS.   (Unwind Error)
    What should I do? 1
>>> Step 6: VERBAL-INVERT-OPERATOR
   I need to take 15.90 and PLUS 66...

--- Options ---
1. 15.9  +  66 = 81.90 (correct with situational support)
2. 15.9  +  66 = 16.56 (bug)
3. 15.9  +  66 = 67.59 (slip)
    What should I do? 1
>>> Step 7: SITUATED-ARITHMETIC
   15.90  +  66 = 81.90

>>> Step 8: VERBAL-KNOW-EVERYTHING-DONE
```

To meet the Step Sufficiency Constraint, we designed EAPS1 to not only model steps in correct strategies, but also common student errors. We modeled two types of errors: arithmetic and conceptual. Conceptual errors include forgetting to invert an operator sign in the verbal representation (see option 2 prior to Steps 3 and 6) or confusing the order of operations in the symbolic representation. For arithmetic errors, we modeled the error of miss-aligning the decimal places in doing arithmetic (see option 2 before Steps 4 and 7) and slips (e.g., 2 * 3 = 5). Because arithmetic was not a focus of our modeling effort, arithmetic bugs and slips are modeled by a single production each (abstracting over detailed arithmetic errors, such as carry errors and borrowing from zero). In EAPS1, each error was associated with a specific "buggy" production (c.f., VanLehn, 1990).

EAPS1 met our first two modeling constraints reasonably well**.** The Solution Sufficiency Constraint was satisfied in that EAPS1 could solve all problems like those in Figure 1. The Step Sufficiency Constraint was satisfied in that EAPS1 produced the kinds of steps we see in the solutions of human problem solvers, as illustrated in the example (MacLaren & Koedinger 1996 provides other examples of the qualitative fit of EAPS1 traces with student solutions).

3. MEETING THE CHOICE MATCHING (C3) AND COMPUTATIONAL PARSIMONY (C4) CONSTRAINTS

The initial development of EAPS1 allowed us to match the model against the wide variety of student solutions, both correct and incorrect, we had observed in the two DFA studies. This initial work gave us insight into the kinds of correct and incorrect knowledge students may have, but it not did give us insight into the strength of students' correct and incorrect knowledge, nor did it explain how students (implicitly) chose between competing strategies. Thus we set out to model student choice processes and make predictions about the frequencies of alternate strategy and error behaviors that result from these internal choices. In other words, we set out to satisfy the Choice Matching Constraint.

3.1 Fitting EAPS1 to Student Strategy and Error Frequencies

To aid in fitting our EAPS models to the DFA data, we segmented the data into a small number of coarse error and strategy categories. Student solutions for result-unknown problems were coded into 4 categories: correct, arithmetic error, conceptual error and no-answer (shown as the major columns in Table 3a). Six types of result-unknown problems, shown in the rows of Table 3a, are determined by crossing two factors: number difficulty (integer vs. decimal) and representation (story vs. word vs. equations). For of these problem types we computed the frequency of the error codes in students' solutions. These frequencies are shown as percentages in the sub-columns labeled "data" in Table 3 (the sub-columns labeled "dif1" and "dif2" are model prediction deviations for EAPS1 and EAPS2, which will be described later). Note that the sum of the data frequencies in each row of Table 3a add to 100% since every solution gets coded into one of these categories. The coding of all but the conceptual error category is straightforward. The conceptual error category includes all solutions that were otherwise categorized**.**

For each of the six types of the more algebra-like start-unknown problems, we not only coded correctness and the three broad error categories, but also a broad strategy coding. This broad strategy coding identifies when solutions involve a formal strategy, that is the use of an algebra equation, versus an informal strategy, either guess-and-test or unwind. The major columns of Table 3b show the combined strategy-error categories for the two strategy codes and four error codes (correct, arithmetic error, and conceptual error codes are separated into those occurring within an informal vs. formal strategy). As in Table 3a, the data frequencies in the rows of Table 3b sum to 100%.

We used ACT-R's utility-based conflict resolution mechanism to model student choice processes and make predictions about consequent strategy and error frequencies. Associated with each production rule is a parameter for the estimated utility of that production. With these parameters set, an ACT-R model will make choices on its own, unlike the hand-selected choices illustrated in Table 2. Productions with higher estimated utilities are more likely to fire. Utilities determine the probability that a production will fire in a given circumstance. When these probabilities are chained together they lead to predictions about the frequencies of associated events (strategy selections or errors). In fitting the model to data, the challenge is to find utility values that yield frequency predictions that match student data.

Table 3. DFA Data, Model Predictions, and Differences for EAPS2.

a. Result-unknown (arithmetic) problems.

Representation	Correct			Arithmetic Errors			Conceptual Errors			No Answer		
	data	dif1	dif2	data	dif1	dif2	data	dif1	dif2	data	dif1	dif2
Integer Story	77	-2	3	1	9	3	17	-10	-9	5	0	3
Integer Word	84	-9	-4	5	5	-1	5	2	3	7	-2	1
Integer Equation	65	-1	-7	7	1	-4	12	-5	5	16	9	6
Decimal Story	63	5	2	17	-1	1	11	-4	-3	9	-4	-1
Decimal Word	42	4	6	36	2	0	21	-14	-13	0	5	8
Decimal Equation	33	7	2	24	8	2	9	-2	8	33	-9	-11

b. Start-unknown (algebra) problems.

	Informal Strategy									Formal Strategy									No Answer		
	Correct			Arithmetic Errors			Conceptual Errors			Correct			Arithmetic Errors			Conceptual Errors			No Answer		
Rep.	data	d1	d2	data	d1	d2	data	d1	d2	data	d1	d2	data	d1	d2	data	d1	d2	data	d1	d2
Integer St	64	-4	-6	2	6	1	14	-5	4	6	1	-4	0	1	0	2	-1	0	14	-1	4
Integer Wd	70	-10	-12	0	8	3	19	-10	-1	0	7	2	0	1	0	2	-1	0	9	4	9
Integer Eq	35	0	0	0	5	2	19	-16	-1	19	0	-17	0	2	0	9	-7	13	19	13	4
Decimal St	45	10	2	10	3	3	24	-16	-6	4	3	-3	0	2	0	1	0	1	15	-2	4
DecimalWd	27	10	8	9	21	17	30	-24	-12	6	-1	-5	0	4	1	9	-8	-8	18	-5	1
Decimal Eq	12	10	9	6	12	9	18	-15	0	6	6	-5	0	10	1	15	-13	-1	42	-10	-12

3.1.1 Setting Parameters in EAPS1

Parameter setting in mathematical models is typically done using an iterative gradient-descent algorithm and depends on fast computations of model predictions given any vector of parameter values. Parameter setting with an ACT-R model is made more challenging because computations of ACT-R model predictions are not as simple as evaluating mathematical expressions. Rather, they involve interpretation of production rules. More importantly, the stochastic nature of ACT-R means the model has to be run multiple times (e.g., 200) on each problem category in order to obtain an estimate of the model's average output under a given parameter setting. Thus an iterative approach applied directly to an ACT-R model may not be practically feasible in many cases.

In fitting parameters for EAPS1 we developed an alternative "incremental complexity" strategy. We started by setting the parameters for the simplest problem category, in other words, the problem category requiring the fewest productions. The columns in Table 4 show the twelve problem categories and the rows show the key productions in the model.[5] The X's in Table 4 indicate which productions are relevant for which problem categories. Note that only 3 productions are relevant to the first two problem categories, "Arith Int Story" and "Arith Int Word." Thus, we started parameter fitting by finding utility values for these three productions that yield choice predictions that best match the error frequencies in the student data (rows 1 and 2 of Table 4). The parameter values selected resulted in frequency predictions of 75% correct, 10% arithmetic errors, 7% conceptual errors, and 5% no answer for the two problem categories (the predictions are the same since the same three productions are involved). These percentages are inferable from Table 3 by adding the "data" and "dif1" columns in the first two rows ("Integer Story" and "Integer Word").

[5] The productions in Table 2 are actually from EAPS2 not EAPS1, but the parameter fitting strategy is just as well illustrated. Details of the EAPS1 productions can be found in MacLaren and Koedinger, 1996.

Table 4. A Skill Matrix, Showing Which EAPS Productions are Relevant to Which Problem Categories.

Skills	Parameter Names	Arith Int Story	Arith Int Word	Arith Int Eq	Arith Dec Story	Arith Dec Word	Arith Dec Eq	Alg Int Story	Alg Int Word	Alg Int Eq	Alg Dec Story	Alg Dec Word	Alg Dec Eq
Comprehend and represent problem													
Verbal Comprehension	VC	X	X	-	X	X	-	X	X	-	X	X	-
Symbolic Comprehension	SC	-	-	X	-	-	X	X_{12}	X_{12}	X	X_{12}	X_{12}	X
Symbol Production (Write Equation)	WE	-	-	-	-	-	-	X_{12}	X_{12}	-	X_{12}	X_{12}	-
Operate on Adjacent Numbers	OA	-	-	-	-	-	-	O	O	O	O	O	O
Deal with unknown input													
Decide what to do to both sides	WI	-	-	-	-	-	-	X_{12}	X_{12}	X_{12}	X_{12}	X_{12}	X_{12}
Invert operator / Invert IO	UC	-	-	-	-	-	-	X_{11}	X_{11}	X_{11}	X_{11}	X_{11}	X_{11}
Unwind-Error	UE	-	-	-	-	-	-	O	O	O	O	O	O
Perform arithmetic													
Arithmetic Procedure	AP	X	X	X	X_{11}	X	X	X	X	X	X_{21}	X	X
Situated Arithmetic	AS	-		-	X_{12}	-	-	-	-	-	X_{22}	-	-
Arithmetic Slip	SL	O	O	O	O	O	O	O	O	O	O	O	O
Arithmetic Bug	BG	-	-	-	O	O	O	-	-	-	O	O	O

After setting these first three parameters, we next set the parameters for the new productions introduced in each slightly more complex group of problems, introducing one new difficulty factor at a time, moving roughly left to right in Table 4. At each step the parameter settings determined at the prior step were fixed and maintained. New settings were determined only for the new parameters needed to address the introduced difficulty. This incremental complexity strategy does not guarantee an optimal fit of the data overall as a better fit may be achieved on more complex problems by slightly reducing the fit to simpler problems. However, the strategy appears to work well; the fit we achieved is good.

In ACT-R, the probability of selecting a production from the conflict set of applicable productions is determined by the relative expected gains of those productions. As described above, expected gain is determined by the expression PG–C, where P is the estimated probability of success if that production fires, G is the estimated value of the current goal, and C is the estimated cost of that production. For each parameterized production, we decided on a cost value (C) and fixed it. We then set the value of P by the parameter fitting strategy described above. These two values, P and C, were then fed into an equation that determines the probability of selecting a production given the other productions in the conflict set. The probability of selecting a given production with expected gain g_i from n possibilities is determined by the Likelihood Ratio equation:

$$p(firing(\mathrm{Pr}\,od_i)) = \frac{e^{g_i/t}}{\sum e^{g_n/t}}$$

where the parameter t is sqrt(6)σ/Π, with σ being the standard deviation of the noise (Anderson & Lebiere 1998, p. 65) and summation is across the n productions in the conflict set.

Our goal is to estimate the parameter P for the productions in the EAPS model to match our DFA data. EAPS1 had 13 parameters: six for strategy selection, three for formal algebra skills (one for solving a verbal problem with a formal strategy (WE), one for manipulating an equation (WI), and one for order-of-operations errors (OA)), one for operator and argument inversion (UC), and three for arithmetic (correct, correct arithmetic on story problems, and arithmetic bugs).[6]

The results of our parameter tuning can be seen in Table 3. The comparison is presented as sets of triples: first the DFA data (in the column labeled "data"), then the EAPS1–DFA difference (in the column labeled "dif1" and "d1"). Table 3a shows the results for arithmetic (result-unknown) problems. Table 3b shows the results for algebra (start-unknown) problems broken down into formal and informal strategies. EAPS1 a good job of capturing the effects of the three difficulty factors on student error and strategy selection behavior. This is illustrated with the model predictions (shown as deviations from the data) for 66 data points in Table 3. 86% (57/66) of the data points for EAPS1 are within 10% of the DFA data. The r^2 value for EAPS1 was 0.92 using 13 parameters.

3.2 Limitations of EAPS1 and the Development of EAPS2

The major empirical weakness of EAPS1 was qualitative. It under-predicted the frequency of conceptual errors on algebra problems. In Table 3, notice the negative values under the two columns labeled "Conceptual Errors" and "d1". As can be seen for both informal and formal strategies, the EAPS1 model consistently predicted fewer conceptual errors than students made, often more than 10% fewer. This observation led us to more closely inspect how both our model and students produce conceptual errors. EAPS1 only made conceptual errors when it fired specific "buggy" productions (e.g., performing addition before multiplication). However, now informed by the model-data mismatch, we hypothesized that students may be making conceptual errors not just because of having buggy knowledge, but also because of lacking correct knowledge. Lack of knowledge might lead students to give up in the middle of a problem when they do not know or are not confident about what to do next. We modified EAPS1 so that it would give up in this way whenever it reached a choice point for which it lacked an appropriate production of sufficient estimated utility (i.e., the gain for achieving the goal is lower than the cost of executing the production). We discuss below the positive consequences of this change on the model-data fit.

The process above illustrates how an effort to meet the Choice Matching Constraint led to a significant change in the model. A second change in EAPS1 resulted not from human data, but from an analytic process of inspecting the computational characteristics of the model. In working with the model, we began to notice certain productions that did not seem necessary. We made an effort to eliminate such productions from the model. Our Computational Parsimony Constraint suggests that eliminating productions that have no computational role and no qualitative or quantitative predictive value leads a simpler model, one that provides better explanations and is more likely to generalize.

In particular, we noticed that the explicit strategy selection productions in EAPS1 were not computationally necessary. Instead, it was possible to simply start the model with productions that implement the first step in each strategy. Thus, for instance, instead of having a first explicit strategy selection production that, in essence, says, "I am going to do algebra" and then a second strategy execution production that starts to translate words into symbol, we found it was possible to start directly with the strategy execution. Besides yielding a simpler model, this change also conformed with our sense that students are not explicitly choosing strategies, particularly, the informal strategies like unwind and guess-and-test, but instead are just responding to the demands of the problem. We can code their consistent behaviors with strategy labels, but it is not necessary that students be explicitly aware of these strategies in order to behave in accord with them (c.f., Alibali & Koedinger, 1999). In fact, students do not easily articulate the steps they are taking and do not label them with strategy names.

The two changes to EAPS1 described above, allowing give ups at any choice point and eliminating explicit strategy selection, led to the creation of EAPS2. As shown in Table 4 to the right of the production names, EAPS2 had 11 parameters corresponding with 11 key productions. Four of the parameterized productions (VC, SC, WE, OA) were for comprehending verbal or symbolic problem statements and creating a quantitative network representation (internally or externally), three were for

---

[6] In EAPS1 the Give-Up, Unwind-Error, and Arithmetic-Slip did not need to be parameterized because their likelihood was one minus the other alternatives.

dealing with an unknown input (WI, UC, UE), and four were for performing arithmetic (AP, AS, SL, BG).

The change from 13 parameters in EAPS1 to 11 in EAPS2 involved some deletions and additions of productions. Six strategy selection productions in EAPS1 were replaced with two parameters (one for verbal and one for symbolic representations) resulting in a decrease of parameters. However, two parameters were added – one for a unwind and one for arithmetic. The additional parameter for arithmetic and unwind in EAPS2 was due to the fact that EAPS2 could give up at every choice point. In EAPS1, p(Unwind-Error) = 1 – P(Unwind), because giving up was the only way to produce and error. In EAPS2, the probability of giving up was 1 – P(Unwind) – P(Unwind-Error).

3.3 Fitting EAPS2 to Student Strategy and Error Frequencies.

To set the parameters for the EAPS2 we created an explicit mathematical model in a Microsoft Excel spreadsheet that corresponds to the behavior of the EAPS2 ACT-R model. As with EAPS1, the inputs to the model are the P values in the PG–C expected gain formula. The resulting expected gains are then input into the Likelihood Ratio equation that produces a set of choice-point percentages for the parameterized productions. These choice-point percentages are used in a set of equations that correspond to different paths that the production rule model can take. These equations predict the frequency percentages for the broad error and strategy categories in Table 3. For each of the 6 kinds of arithmetic problems, we used the ACT-R model to guide us in writing 4 equations that compute the probability of each of 4 broad categories of errors on arithmetic problems: no error, arithmetic errors, conceptual errors and no answer. For each of the 6 kinds of algebra problems we used the ACT-R model to guide us in writing 7 equations that compute the probability of each of the 7 strategy and error categories (no error, arithmetic errors, and conceptual errors for informal and formal strategies, and no answer). We wrote a total of 66 equations, 6x4=24 for arithmetic problems and 6x7=42 for algebra problems (these equations are shown in Appendix 1).

Each equation for a broad error category is in the form of a sum of products, where each product term corresponds to a different path through the model that results in the same broad error category. For example, one of the terms in the equation for arithmetic errors for integer verbal arithmetic is VC*AR*VC*SL. This represents the path to perform the first verbal comprehension (VC) and arithmetic operation (AR) correctly but after the second quantitative relation is comprehended (VC), the model performs an arithmetic slip (SL). To find the utility parameters that best fit EAPS2 with the DFA data we used Excel's Solver tool.

Because most of the productions had multiple contexts in which they could fire, determining the utility values for each production was not simply a matter of determining the probability that that production would fire. For example, for arithmetic problems there are three different contexts for the Arithmetic Procedure. In all cases, the utility value of this production is 16.38 (out of a total of 20), but the probability that this production will fire depends on the context (i.e., what other productions are competing with it). As can be seen in Table 4, for integer arithmetic problems (column 1-3) there is one production that competes with Arithmetic Procedure, namely Arithmetic-Slip. In this case, the probability of Arithmetic Procedure (AP_e) is 97% (computed by plugging 16.38 and Arithmetic-Slip's expected gain of 14.36 into the choice formula in section 3.1.1). For decimal arithmetic word problems and equations, there are two productions that compete with Arithmetic Procedure: Arithmetic-Slip and Arithmetic-Bug. In this case the probability of Arithmetic Procedure (AP_h) is 76%. For decimal arithmetic story problems, in contrast, there are three productions that compete with Arithmetic Procedure: Situated-Arithmetic, Arithmetic-Slip, and Arithmetic-Bug. In this case, the probability of Arithmetic Procedure (AP_s) is 37%.

Table 5 shows the eighteen math model equations that correspond with alternative paths to correct answers to the 6 main problem categories. The first math model equation is for integer arithmetic on result-unknown story problems and involves two applications of verbal comprehension (VC_1) (one for each quantitative relation) and two of arithmetic for integer problems (AP_e). For integer arithmetic word problems (see the second equation) we can see that the math model predicts the same performance as integer arithmetic story problems, but for the integer arithmetic equations (see the third math model equation) the comprehension component is different: symbol comprehension (SC) replaces verbal comprehension (VC). For the decimal arithmetic problems there is a difference between story and word problems. For the story problems, a situated arithmetic process (AS) also applies, models the cognitive process involved of using situated knowledge of the money units to support the appropriate alignment of decimal points in performing arithmetic (i.e., AS models knowing not to add dollars to cents).

For correct performance of the informal unwind strategy the only additional skill that is required is the ability to unwind correctly (UC), which is implemented at the productions Invert-Operator and Invert-Input-Output. This example illustrates how parameters in the math model represent the aggregation of sets of productions in the ACT-R model that always fire together.

For more details on how the cognitive model was used to create the math model see Appendix 1.

Table 5. Math Model Equations for Paths Leading to Correct Performance.

Integer Result-unknown:		
Story	$VC_1(.92)$ * APe(.97) * $VC_1(.92)$ * APe(.97) (resulting percent correct)	= 80%
Word	$VC_1(.92)$ * APe(.97) * $VC_1(.92)$ * APe(.97)	= 80%
Equation	$SC_1(.78)$ * APe(.97) * $SC_1(.78)$ * APe(.97)	= 58%
Decimal Result-unknown:		
Story	$VC_1(.92)$ * (APs(.37) + AS(.52)) * $VC_1(.92)$ * (APs(.37) + AS(.52))	= 65%
Word	$VC_1(.92)$ * APh(.76) * $VC_1(.92)$ * APh(.76)	= 48%
Equation	$SC_1(.78)$ * APh(.76) * $SC_1(.78)$ * APh(.76)	= 35%
Integer Start-unknown (Informal Strategy):		
Story	$VC_2(.88)$ * UC(.87) * APe(.97) * $VC_1(.92)$ * UC(.87) * APe(.97)	= 58%
Word	$VC_2(.88)$ * UC(.87) * APe(.97) * $VC_1(.92)$ * UC(.87) * APe(.97)	= 58%
Equation	$SC_2(.61)$ * UC(.87) * APe(.97) * $SC_1(.78)$ * UC(.87) * APe(.97)	= 35%
Decimal Start-unknown (Informal Strategy):		
Story	$VC_2(.88)$ * UC(.87) * (APs(.37) + AS(.52)) * $VC_1(.92)$ * UC(.87) * (APs(.37) + AS(.52))	= 47%
Word	$VC_2(.88)$ * UC(.87) * APh(.76) * $VC_1(.92)$ * UC(.87) * APh(.76)	= 35%
Equation	$SC_2(.61)$ * UC(.87) * APh(.76) * $SC_1(.78)$ * UC(.87) * APh(.76)	= 21%
Integer Start-unknown (Formal Strategy):		
Story	WE(.04) * $SC_2(.61)$ * UC(.87) * APe(.97) * $SC_1(.78)$ * UC(.87) * APe(.97) +	
	WE(.04) * $WI_2(.18)$ * $SC_2(.61)$ * APe(.97) * $WI_1(.52)$ * $SC_1(.78)$ * APe(.97)	= 1.6%
Word	WE(.04) * $SC_2(.61)$ * UC(.87) * APe(.97) * $SC_1(.78)$ * UC(.87) * APe(.97) +	
	WE(.04) * $WI_2(.18)$ * $SC_2(.61)$ * APe(.97) * $WI_1(.52)$ * $SC_1(.78)$ * APe(.97)	= 1.6%
Equation	$WI_2(.18)$ * $SC_2(.61)$ * APe(.97) * $WI_1(.52)$ * $SC_1(.78)$ * APe(.97)	= 1.5%
Decimal Start-unknown (Formal Strategy):		
Story	WE(.04) * $SC_2(.61)$ * UC(.87) * (APs(.37)+AS(.52)) * $SC_1(.78)$ * UC(.87) * (APs(.37)+AS(.52)) +	
	WE(.04) * $WI_2(.18)$ * $SC_2(.61)$ * (APs(.37)+AS(.52)) * $WI_1(.52)$ * $SC_1(.78)$ * (APs(.37)+AS(.52))	= 1.3%
Word	WE(.04) * $SC_2(.61)$ * UC(.87) * APh(.76) * $SC_1(.78)$ * UC(.87) * APh(.76) +	
	WE(.04) * $WI_2(.18)$ * $SC_2(.61)$ * APh(.76) * $WI_1(.52)$ * $SC_1(.78)$ * APh(.76)	= 0.9%
Equation	$WI_2(.18)$ * SC_2 (.61) * APh(.76) * $WI_1(.52)$ * $SC_1(.78)$ * APh(.76)	= 0.9%

Model-Data Fit for EAPS2

The results of our parameter tuning using the math model for EAPS2 can be seen in Table 3. The comparison is presented as sets of triples: first the DFA data, then the EAPS1–DFA difference, and then the EAPS2–DFA difference.

Both models do a good job of capturing the effects of the three difficulty factors on student error and strategy selection behavior. 88% (58/66) of the data points for EAPS2 are within 10% of the DFA data (compared with 86% of the data points for EAPS1). The r^2 value for EAPS2 was 0.95 using 11 parameters (compared with 0.92 for EAPS1 using 13 parameters).

Because EAPS2 could give up at any choice point in the solution it was possible to improve the under-prediction of conceptual errors in EAPS1. Unlike EAPS1, the predictions of EAPS2 are not systematically different from the data (see the informal conceptual error column in Table 3b).

In summary, EAPS2 achieved an equivalent quantitative fit as EAPS1 with fewer parameters, in a more intuitive fashion, and without the systematic deviations from the error data. The difference between the two models yielded two important lessons for our theory of early algebra problem solving. First, we do not need to assume that students are making explicit strategy choice and, in fact, it is more parsimonious to assume that their strategy choices are implicit. Second, in contrast to accounting for all student errors as systematic bugs, we found that a large proportion of student errors can be accounted for by lack of knowledge.

4. MEETING ACQUIRABILITY (C5) AND TRANSFERABILITY (C6) CONSTRAINTS

EAPS2 eliminated the requirement for explicit strategies and bugs while actually improving the fit with our DFA data, but it was still a fairly coarse grained model. For example, complex skills were sometimes represented as single productions. After developing EAPS2, we asked: Does the model make sense on a finer production rule level? What predictions does it make about the learning process? We wanted to focus more intently on the plausibility of the production rules, and to provide a sound qualitative account of how these productions could be acquired and tuned. Our aim is to not only create a model that describes the quantitative character of problem solving, but also provides a process explanation, a mechanism, for individual steps in problem solving behavior (c.f., Newell & Simon, 1972, p. 9). We also wanted the model to characterize different states of learning and be able to make predictions about the learning process.

We start our description of EAPS3 with examples of it in operation. Next we describe how we applied the Acquirability and Transfer constraints in developing EAPS3. We end this section with some predictions of our theory of early algebra problem solving that follow from the changes made to create EAPS3.

4.1. An Example-Based Overview of EAPS3

Table 6 shows a trace of EAPS3 using Unwind on the Decimal Start-Unknown Story problem shown in Figure 3. At the first choice point, the model considers three possibilities: 1) focusing on the last relation in a verbal representation (which has a 99.8% chance of being selected), 2) translating the verbal representation into a symbolic form (WE), that is, writing an equation (which has a 0.2% chance of being selected), and 3) giving up, returning no answer (NA) (which has a 0% chance of being selected). The most likely alternative is chosen, and in Step 1 the model begins by focusing on the relation All-Donuts-Price[7] Divided-by Donut-Price gives Donut-Number.

Next, in Step 2, EAPS3 forms an arithmetic goal using the numbers in the relation it is focused on, in this case 0.37 and 7. After deciding which relation and which arguments to act on, in Step 3 the model decides how to reason with a relation with an unknown input, in this case setting a goal to unwind the operator. In steps 4 and 5, this goal is achieved by inverting the operator (divide becomes multiply) and inverting the output and the second input. The resulting arithmetic problem is performed in Step 6. Here the model writes down 7 TIMES 0.37, and produces 2.59 as an answer. This corresponds to the first part of the student's column arithmetic in Figure 3. Producing the value 2.59 for All-Donuts-Price completes the processing for the first relation. The second relation is processed in steps 7-12 as the model works backwards though the "subtracts $0.22 for the box they came in" relation. After solving the second relation a production recognizes the problem is done (Step 13).

[7] The label "All-Donuts-Price" is for our purposes in reading and understanding the model. It does not imply that students have labels for such intermediate quantities.

Table 6. EAPS3 unwinding a story problem

```
--- Options ---
   1. 99.8%     Focus On (All-Donuts-Price Divided-by Donut-Price -> Donut-Number)
   2.  0.2%     WE: X - 0.22 / 0.37 = 7                      (Write an equation)
   3.  0.0%     NA:  I'll just give up.
What should I do? 1
>>> Step 1: Focus-On-Last-Relation
   1. 88.8%     VC: All-Donuts-Price Divided-by 0.37 is 7..(Form an arithmetic goal)
   2.  4.2%     WE: X - 0.22 / 0.37 = 7                      (Write an equation)
   3.  7.0%     NA:  I'll just give up.
What should I do? 1
>>> Step 2: Form-Arithmetic-Goal-For-Verbal                  VC
>>> Step 3: Set-Goals-To-Unwind                              UC
   1. 86.6%     UC:  So DIVIDED-BY becomes TIMES.            (Invert operator)
   2. 13.4%     UE:  So DIVIDED-BY ...                       (Operator Inversion Error)
What should I do? 1
>>> Step 4: Invert-Operator                                  UC
   So DIVIDED-BY becomes TIMES
>>> Step 5: Invert-Input-Output                              UC
   Some number ... 0.37 =7 ... 7 TIMES 0.37
   1. 96.6%     AP:  7  TIMES  0.37 = 2.59        (Normal Arithmetic)
   2.  2.0%     SL:  7  TIMES  0.37 = 25.9        (Arithmetic Slip)
   3.  1.4%     AP:  I can't do this...
What should I do? 1
>>> Step 6: Perform-Arithmetic                               AP
   7  TIMES  0.37 is 2.59
   1. 100%      FoR: Focus On (Total-Amount Minus Box-Price -> All-Donuts-Price)
   2. 0.0%      WE: X - 0.22 = 2.59                          (Write an equation)
   3.  0.0%     NA:  I'll just give up.
What should I do? 1
>>> Step 7: Focus-On-Last-Relation
   1. 88.8%     VC: TOTAL-AMOUNT MINUS 0.22 is 2.59. (Form an arithmetic goal)
   2.  4.2%     WE: X - 0.22 = 2.59                          (Write an equation)
   3.  7.0%     NA:  I'll just give up.
>>> Step  8: Form-Arithmetic-Goal-For-Verbal                 VC
>>> Step  9: Set-Goals-To-Unwind                             UC
   1. 86.6%     UC:  So MINUS becomes PLUS.       (Invert operator)
   2. 13.4%     UE:  So MINUS ...                 (Operator Inversion Error)
What should I do? 1
>>> Step 10: Invert-Operator                                 UC
   So MINUS becomes PLUS
>>> Step 11: Invert-Input-Output                             UC
   Some number ... 0.22 =2.59 ... 2.59 PLUS 0.22
   1. 96.6%     AP:  2.59  PLUS  0.22 = 2.81                 (Normal Arithmetic)
   2.  2.0%     SL:  2.59  PLUS  0.22 = 28.1                 (Arithmetic Slip)
   3.  1.4%     AP:  I can't do this...
What should I do? 1
>>> Step 12: Perform-Arithmetic                              AP
   2.59  PLUS  0.22 is 2.81
>>> Step 13: Vrb*Have-Answer*Know-Everything-Correct
   So the answer is 2.81.  (Correct)
```

Now let's look at EAPS3 using formal algebra to solve the story problem in Figure 2. A trace of this problem being solved using formal algebra is shown in the trace in Table 7.[8] The model starts off by writing an equation and then focusing on the last relation in the problem, in which Laura adds 10 cents to get the total price. The production Write-Inverse-On-Both-Sides (WI) then inverts that last operation and sets a goal to write a new equation. It then writes an equation and re-parses it (i.e., it produces an internal network of relations reflecting the external representation of the equation).

The part of the model that parses an equation (Steps 5 through 14) consists of productions that comprehend numbers, operators, and variables, as well as productions that recognize that two terms connected by an operator produce a new quantity which can then serve as the input to another relation.

Table 7. EAPS3 performing formal algebra on a story problem (Fig 2)

```
>>> Step  1: Write-Equation-For-A-Verbal-Problem                    WE
   Write equation: (X * 25 + 10 = 110)
>>> Step  2: Focus-On-Last-Relation
   Focus on ALL-DONUTS-PRICE PLUS BOX-PRICE TOTAL-PRICE
>>> Step 3: Write-Inverse-On-Both-Sides                             WI
   So write … 10 on both sides…
>>> Step 4: Invert-Operator                                         SC/VC
    So PLUS becomes MINUS
>>> Step  5: Write-Equation                                         WE
   Write new equation:  (X * 25 + 10 – 10 = 110 - 10)
>>> Step  6: Comprehend-Variable                                    SC
>>> Step  7: Comprehend-Operator                                    SC
>>> Step  8: Comprehend-Number                                      SC
>>> Step  9: Comprehend-Quantity-Operator-Quantity-High-Precedence  SC
   [X * 25 -> Q1  (Relation1) In  (X * 25)
>>> Step 10: Comprehend-Equals-Sign                                 SC
>>> Step 11: Comprehend-Number                                      SC
>>> Step 12: Comprehend-Operator                                    SC
>>> Step 13: Comprehend-Number                                      SC
>>> Step 14: Comprehend-Quantity-Operator-Quantity-Low-Precedence   SC
   [110 - 10 -> Q2  (Relation2) in  (X * 25 = 110 - 10)
>>> Step 15: Comprehend-Lefthand-Side-Equals-Righthand-Side         SC
   Q1 = Q2  In  (X * 25 = 110 - 10)
>>> Step 16: Focus-On-Last-Relation                                 SC
>>> Step 17: Form-Arithmetic-Goal-for-Equation                      SC
   Simplify - 10 = Q1  in:  (X * 25 = 110 - 10)
>>> Step 18: Interpret-Operator
    So - ... is ... minus
>>> Step 19: Perform-Arithmetic                                     AP
    110  minus  10 is 100
>>> Step 20: Update-Equation
>>> Step 21: Focus-On-Last-Relation
>>> Step 22: Write-Inverse-On-Both-Sides                            WI
   So write … 25 on both sides…
>>> Step 23: Invert-Operator                                        SC/VC
   So TIMES becomes MULTIPLY
>>> Step 24: Write-Equation                                         WE
   Write new equation:  (X * 25 / 25  = 100 / 25)
>>> Step 25: Comprehend-Variable                                    SC
…
>>> Step 29:Comprehend-Quantity-Operator-Quantity-High-Precedence   SC
   [100 / 25 -> Q3  (Relation3) In  (X = 100 / 25)
```

[8] In these traces the model is "run" rather than walked; other options are not shown, the model just selects the most likely production and fires it.

```
>>> Step 30.: Comprehend-Lefthand-Side-Equals-Righthand-Side        SC
   X = Q3  In  (X = 100 / 25)
>>> Step 31: Focus-On-Last-Relation                                 SC
>>> Step 32: Form-Arithmetic-Goal-for-Equation                      SC
   Simplify / 25 = X  In:  (X = 100 / 25)
>>> Step 33: Interpret-Operator1
   So / ... is ... divide
>>> Step 34: Perform-Arithmetic                                     AP
   100  Divide  25 Is 4
>>> Step 35: Have-Answer-For-Equation-Correct
            So the answer is 4.  (Correct)
```

Finally, let's look at an example of EAPS3 making an Order-of-Operations error. An example of an order of operations error is shown in Figure 7, and the model's solution is shown in Table 8.

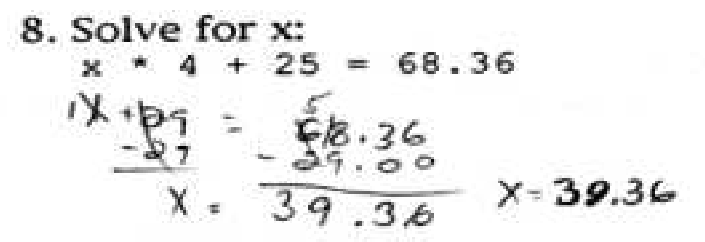

Fig. 7. An Order-of-Operations Error

Order-of-Operations bugs in EAPS3 are modeled by an overly general production that simply acts on two integers side by side. EAPS3 parses the equation in Step 6, immediately operating on the adjacent numbers "4 + 25," instead of completely comprehending the entire equation. The model then finishes comprehending the rest of the equation and performs the second algebra step correctly in Step 14. It then finishes the problem with the incorrect answer.

Table 8. EAPS3 making an order-of-operations error on an equation: X*4+25=68.36

```
>>> Step  1: Comprehend-Variable
>>> Step  2: Comprehend-Operator
>>> Step  3: Comprehend-Number
>>> Step  4: Comprehend-Operator
>>> Step  5: Comprehend-Number
>>> Step  6: Operate-On-Adjacent-Numbers-Then-Subtract              OA
   OO:  Simplify 4 + 25  In S4E-Eq:  (X + Result$1695)
>>> Step  7: Interpret-Operator
    So + ... Is ... Plus
>>> Step  8: Perform-Arithmetic                                     AP
   4  Plus  25 Is 29
>>> Step  9: Comprehend-Equals-Sign
>>> Step 10: Comprehend-Number
>>> Step 11: Comprehend-Quantity-Operator-Quantity-Low
   [X + 29 -> Res$1696  (Rel$1694) In  (X + 29 = 68.36)
>>> Step 12: Comprehend-Lefthand-Side-Equals-Righthand-Side
   68.36 = 68.36  In  (X + 29 = 68.36)
>>> Step 13: Focus-On-Last-Relation
>>> Step 14: Write-Inverse-On-Both-Sides
>>> Step 15: Write-Equation1                                        WE
   New-Eq:  (X "=" 68.36 - 29)
>>> Step 16: Comprehend-Variable
>>> Step 17: Comprehend-Equals-Sign
>>> Step 18: Comprehend-Number
>>> Step 19: Comprehend-Operator
```

```
>>> Step 20: Comprehend-Number
>>> Step 21: Comprehend-Quantity-Operator-Quantity-Low
   [68.36 - 29 -> Res$1698  (Rel$1697) In  (X = 68.36 - 29)
>>> Step 22: Comprehend-Lefthand-Side-Equals-Righthand-Side
   X = Res$1698  In  (X = 68.36 - 29)
>>> Step 22: Focus-On-Last-Relation
>>> Step 23: Form-Arithmetic-Goal-for-Equation                SC
   Simplify - 29 = X  In:  (X = 68.36 - 29)
>>> Step 24: Interpret-Operator1
    So - ... Is ... Minus
>>> Step 25: Perform-Arithmetic                               AP
   68.36  Minus  29 Is 39.36
>>> Step 26: Sym*Have-Answer* Wrong
```

4.2 How we developed EAPS3

4.2.1 How we met the Acquirability Constraint

The development of EAPS3 was constrained not only by the first four constraints (C1-C4), but also by the final two constraints (C5, C6) regarding learning, Acquirability and Transfer. We discuss how we modified EAPS2 to meet these constraints.

The Acquirability Constraint asks whether the productions in a model could be plausibly acquired. The sole mechanism for acquiring new productions in ACT-R is analogy. Productions acquired through analogy may or may not be correct. However, such productions are always correct at least in contexts like those in which they are acquired. This Acquirability Constraint was not met in EAPS2. For instance, the buggy Order-of-Operations production in EAPS2 specifically looked for instances where operator precedence rules could be explicitly violated, for example choosing to operate on “4+3” in the expression “5*4+3” because “*” has higher precedence than “+.” This production violates the Acquirability Constraint because there is no context in which this production produces correct behavior.

The old productions that implemented order of operations bugs in EAPS2 were never correct under any circumstances. In EAPS3, order of operations errors have been implemented by the production Operate-on-Adjacent-Numbers[9] that sometimes generates correct behavior but sometimes does not. Operate-on-Adjacent-Numbers can fire when two numbers with an operator in between are seen in an equation. For instance, in the equation “X + 4 * 25 = 115” it could fire and correctly produce “X + 100 = 115.” However, in the equation “X * 4 + 25 = 115” it can also fire, but it incorrectly produces “X * 29 = 115.” This production is overly-general, working correctly in some circumstances, but missing conditions in its left-hand-side that prevent it from incorrectly applying in other situations.

We modified EAPS3 to meet the Acquirability Constraint by modifying or eliminating any production that could not produce a correct result (and thus could not be learned by analogy). As illustrated above, some error-producing productions were modified to be overly-general. Below we illustrate another way to model errors that meets the Acquirability Constraint, namely, the forgetting of intermediate goals.

4.2.2 How we met the Transfer Constraint

Besides Acquirability, we also wanted EAPS3 to meet the Transfer Constraint (C6). The Transfer Constraint asks whether the model predicts sufficient transfer between problem categories. Upon inspection, it seemed that the transfer predictions for EAPS2 were inadequate. For example, it seems

[9] Actually, there is also a simple variant of Operate-on-Adjacent-Numbers, Operate-On-Adjacent-Numbers-Then-Subtract, shown in the last trace, which performs a normal order-of-operations error, and then subtracts that result from the unknown. This bug was surprisingly common in DFA1, and may result from the high frequency in textbooks of mx+b form problems that reinforce a shallow subtract-divide procedure.

possible that if a student learns to solve equations informally, that some of that acquired knowledge could transfer to solving equations using formal algebra. Some of the skills that might transfer are deciding what part of the equation to operate on and what new operation needs to be performed. EAPS2 did not have an underlying representation that was common to both equations and verbal problems. Thus, such transfer was not possible.

Examining the productions in EAPS2, we noticed a number of similarities between pieces of different processes that were not apparent in our initial coarser grain modeling – they were hidden within long complex production rules. For example, different productions were used to model the unwind (UC) and equation solving (WI) strategies even though there are significant similarities in their key steps. In both cases, students need to 1) find the last operation in a procedure or expression, the one that needs to be inverted, and 2) to invert that operation. The difference is in the representation that gets manipulated. In Equation Solving, students are applying the inverted operation (e.g., subtract 10) to both sides of an equation (e.g., $3x + 10 = 70$). In Unwind, students are applying the inverted operation (e.g., subtract 10) to the result quantity (e.g., 70). This difference is seen in what is written down as illustrated in the contrast between the Equation Solving solution in Figure 2 and the Unwind solution in Figure 3. Despite this difference, both strategies involve finding the operation to invert (e.g., adding ten) and finding the inverse of an arithmetic operator (e.g., subtraction is the inverse of addition).

EAPS2 did not meet the Transfer Constraint because these similarities in processing were not apparent in the model, or more precisely, they were not represented as productions common to the two strategies. Certainly similar processes were being performed but this similarity was buried in two complex productions, Unwind (UC) and Write-Inverse-on-Both-Sides (WI) that implemented the key steps in the Unwind and Equation Solving strategies. To meet the Transfer Constraint in EAPS3, we broke up these single complex productions into several smaller and simpler productions. Table 9 shows the four smaller EAPS3 productions that perform essentially the same process as the single Unwind production in EAPS2. Table 10 shows the four smaller EAPS3 productions (two of which are common to Unwind) that perform essentially the same process as the single Write-Inverse-on-Both-Sides production in EAPS2. The Transfer Constraint is thus met in EAPS3 because now the common steps in these strategies are implemented by the same productions, Focus-on-Last-Relation and Invert-Operator.

Table 9. Productions for Unwind Strategy

1. FOCUS-ON-LAST-RELATION
IF the goal is to solve a problem
and there is a quantitative relationship with an operation
involving an unknown input, and a known input quantity
that produces a known output quantity.
THEN focus on that quantitative relationship.

2. SET-GOALS-TO-UNWIND
IF the focus is on a quantitative relation, and the first input is unknown,
And the second input and the output are known
THEN set goals to invert the operator and to
invert the second input and output.

3. INVERT-OPERATOR
IF there is a goal to invert an operator,
and you know what the inverse of the operator is,
THEN invert the operator and pop the goal.

4. INVERT-INPUT-OUTPUT
IF there is a goal to invert the second input and output of a quantitative relation,
and you know what the output and second input are but not the first input,
THEN combine the output and second input using the operator inverted to get the second.

Table 10. Productions for Interpretive Equation Solving

1. FOCUS-ON-LAST-RELATION (Same as in Table 9)

2. WRITE-INVERSE-ON-BOTH-SIDES
IF the focus of attention is on the last operation in one side of an equation
and that operation involves a known number,
THEN set goals to invert the operator and to
write a new equation with the inverse of the operator and the number on both sides.

3. INVERT-OPERATOR (Same as in Table 9)

4.WRITE-EQUATION
IF there is a goal to write a new equation with an operator and number to add to each side,
THEN write a new equation with that operator and number added to each side.

In order to implement this change in EAPS3, we needed to make some significant changes in our assumptions (and the model's code) about how equation solving is performed. In EAPS2, we modeled equation solving as a largely perceptual-motor process that operated fairly directly on the external representation of equations on paper. The idea is that students "see" what change to make and then rewrite the equation to implement that change. In this approach the model is operating on visual representations of the equation (c.f., Kirshner, 1989). The informal unwind strategy, in contrast, operates on the quantitative network, a verbal representation. To have productions in common between equation solving and informal unwind, we needed these processes to be operating on the same internal representation.

In addition to this computational clue, we were also led to a common internal representation for equation solving and unwind, by our data analysis. Recall the large discrepancy between students' lack of success with formal equation solving (19% no answer for integer start-unknowns see Table 3) versus their relative success with informal strategies on equations (35% correct for integer start-unknowns) and even greater success with informal strategies on word equations (70% correct for integer start-unknowns). If only students "read" and understood an equation just as they read the analogous word equation, they should be able to solve equations much more effectively.

Thus, for both computational and empirical reasons we implemented an alternative approach to equation solving in EAPS3. In addition to the "Visual Equation Solving" strategy of EAPS2, we also implemented an "Interpretive Equation Solving" strategy in EAPS3. This strategy begins by translating the equation into the same quantitative network representation that is created to model the reading of a word-equation – this models the student who is "reading" and "understanding" the equation. In addition to the production changes shown in Tables 9 and 10, we also added productions in EAPS3 to parse external symbolic expressions and create quantities and quantitative relations. Once the quantitative network representation is created, in addition to Visual Equation Solving, EAPS3 can solve the equation either via Unwind or via Interpretive Equation Solving.

Let us consider in more detail the similarities and differences between the productions for Unwind (Table 9) and for Interpretive Equation Solving (Table 10). Both strategies use the productions Focus-on-Last-Relation and Invert-Operator. After Focus-on-Last-Relation, the productions Set-Goals-to-Unwind and Write-Inverse-On-Both-Sides play analogous but different roles in the respective strategies, setting goals for the next two productions to achieve. The Invert-Operator production achieves one of these goals in both strategies. The key difference between these strategies is in the final productions, Invert-Input-Output and Write-Equation. Invert-Input-Output operates on the quantitative network, an internal mental representation, while Write-Equation generates a new equation on paper, an external representation. This difference between internal and external processing is important precisely because a major advantage of equation solving is the ability to off-load working memory by using paper as an external memory store.

The addition of Interpretive Equation Solving leads to the following prediction in EAPS: Successful equation solvers learn to first comprehend the equation and "make sense" of the formal operations in terms of the quantitative relations expressed. Less successful learners only acquire shallow rules that operate more directly on a visual representation of equations (c.f. Kirshner, 1989).

Interpretive Equation Solving uses a deeper *verbal* representation of the equation (the quantitative network) in addition to the external visual representation. Visual Equation Solving is cued only by the external visual representation, without complete comprehension of the quantitative relations. Thus, this strategy is more error prone, susceptible for instance to faulty inferences based on visual proximity, like incorrectly adding 4 and 25 in "x * 4 + 25" (see Figure 7). Even though Interpretive Equation Solving requires more complex processing than the Visual Equation Solving, it builds on existing skills from more familiar verbal problems. Thus, it provides an avenue for students to more quickly and easily acquire robust equation solving with understanding.

Given the efficiencies of direct visual processing, it is likely that expert equation solvers use a process closer to Visual Equation Solving, however, we suspect such productions are acquired through the automization of an earlier interpretive process performed with understanding rather than acquired through direct visual analogies.

4.3 Predictions Resulting from EAPS3 Changes

The central motivation for developing EAPS3 was to better explain the transfer of skills learned in simpler problems to more complex ones. EAPS3 makes some novel predictions about skill transfer. Consider the problem types in our DFA studies (Figure 1). Table 11 shows the main skill sets that apply to these different problem types. The main skills (productions) are listed on the left as rows and different strategies and problem types that these apply to are listed across the top. Moving left to right, each column introduces new skills for the model to acquire.

The skill sets in Table 11 provide a hypothesized developmental sequence for learning early algebra, where new skills are acquired incrementally through careful sequencing of problems (from left to right in the table). In this sequence, according to EAPS3, learning to identify the last operation that produces the result (Focus-on-Last-Relation) would take place during informal algebra on verbal problems (Table 11) and would then transfer to the more complex problems. This Focus-on-Last-Relation skill implies that, once a student can comprehend the meaning of a problem, whether given verbally or symbolically, he or she can then identify the similar operation to undo, either informally or through equation solving. The Invert-Operator skill is useful for both verbal and symbolic problems. In EAPS2, there was no common representation for these productions to apply to, and so this transfer was not possible.

The third column in Table 11 (Set 3) shows the needed skills (and parameter estimate) for an informal strategy applied to a verbal problem (as shown in Table 6). We can see that the main skills for this strategy/problem type combination are Focus-on-Last-Relation, Form-Arithmetic-Goal-for-Verbal (VC), Invert-Input/Output, and Invert-Operator (UC). If the model were then to attempt to solve Verbal problems using formal algebra (Set 4, the fourth column in Table 11), it would be exercising those skills in the context of the symbolic representation, along with productions for equation generation (WE) and manipulation (WI). As shown at the bottom of Table 11, problems in Set 3 are correct 58% of the time, while problems in Set 4 are correct only 35% of the time. This analysis yields the following prediction: informal algebra skills on verbal problems should be practiced to sufficiently strengthen those skills that transfer to informal algebra on equations before moving on to algebra equations. The same is true of skills in Set 4 before moving on to Set 5 and Set 6.

Table 11. Coarse-Grain "Skill Sets" for EAPS3

Productions		Verbal Arith (Set1)	Symbol Arith (Set2)	Informal Algebra on Verbal (Set3)	Informal Algebra on Eq (Set4)	Formal Algebra on Verbal (Set5)	Formal Algebra on Eq (Set6)
Comprehension							
Comprehend verbal representation	(VC)	92%	-	92%	-	92%	-
Comprehend symbolic rep.	(SC)	-	78%	-	78%	78%	78%
Focus on last relation		-	-	100%	100%	100%	100%
Operator Interpretation/Inversion							
Invert input/output	(UC)	-	-	94%	94%	-	-
Invert operator	(UC)	-	-	94%	94%	94%	94%
Formal Algebra							
Write inverse on both sides	(WI)	-	-	-	-	11%	11%
Write equation	(WE)	-	-	-	-	4%	-
Arithmetic							
Integer Arithmetic	(AP_e)	97%	97%	97%	97%	97%	97%
Performance on Two Operator Problems		80%	58%	58%	35%	1.6%	1.5%

The implication of this transfer prediction is significant. According to the math model, the skills for doing formal equation solving (WI) were executed correctly only 11% of the time, while skills implementing unwind (UC) were executed correctly 90% (94% * 94%) of the time for the average student. Because the formal algebra skill is initially very weak, it is important to strengthen the other skills required for implementing it. If our hypothesis about Focus-on-Last-Relation and Invert-Operator transferring to formal algebra is correct, it should be possible to enhance readiness to learn equation solving by practicing the Unwind strategy prior to introducing formal algebra. Testing this prediction is on our agenda of future work, which will be discussed next, in section 5.2.

5. DISCUSSION AND CONCLUSIONS

Like any theory, the EAPS theory is useful for a number of reasons: to provide explanations, to make predictions, and to evaluate novel hypotheses and alternative designs. EAPS provides explanations and predictions about multiple aspects of early algebra problem-solving performance, learning, and instruction. We discuss EAPS explanations of problem difficulty effects and the nature and frequency of student strategies and errors (section 5.1). We also discuss (section 5.2) predictions of the theory regarding early algebra learning and instruction.

5.1 Using EAPS to Explain Prior Results

Theories are appropriately judged by their ability to make novel predictions beyond the scope of the empirical data that drove their creation. However, the ability of a theory to explain prior empirical results is also important and particularly so when the prior results are surprising or counter to common belief. For instance, Einstein's theory of relativity was important not just for its predictions but because it provided an explanation for Michelson and Morley's surprising data on the constancy of the speed of light.

5.1.1 Explanations of Problem Difficulty

The EAPS theory was motivated by the surprising result that word problems are easier than equations for a class of early algebra problems. In addition to statements in the cognitive literature regarding the "notorious difficulty of story problems" (Cummins, Kintsch, Reusser & Weimer, 1988), we confirmed the prevalence of this common belief in formal surveys of mathematics teachers and education researchers (Nathan & Koedinger, 2000). Of course, "surprise" is in the eye of the beholder. Some observers of our results have hypothesized that story problems may be easier because the situational context of the story problem cues relevant real world knowledge. Indeed other studies have shown advantages for story problems under certain circumstances (Carraher, Carraher & Schlieman, 1987; Baranes, Perry & Stigler, 1989) and have hypothesized that the situational cues in these problems support problem solving. However, this explanation is contradicted by our results --students did just as well or better on integer word equation problems (77%) as integer story problems (74%) whereas integer equations were harder (60%). In other words, it is not the story context that makes story problems easier than equations.

So what is the explanation for this verbal advantage? A careful inspection of the EAPS model provides an explanation. Two reasons account for the difference in performance: differences in problem comprehension and differences in strategy selection. By problem comprehension we mean the translation of an external problem representation, whether story, verbal or symbolic, into the internal quantitative network representation of the quantities and relations (see Figure 4). The importance of problem comprehension to explaining the performance difference between word problems and equations is highlighted by the fact that this difference is present even for result-unknown problems.

EAPS accounts for this verbal advantage by differences in the acquired utility of verbal comprehension versus symbolic comprehension productions. This explanation can be derived from the parameter values shown in Table 5. (For simplicity we report the parameters for the integer problems – the results are analogous using the parameters for the real number problems.) For result-unknowns, correct verbal comprehension (VC_1) productions fire 92% of the time while correct symbolic comprehension (SC_1) productions fire only 78% of the time. This difference in comprehension also plays a role in explaining why verbal start-unknowns are easier than symbolic start-unknowns (VC_2 = 88% versus SC_2 = 61%). However, the model suggests a second reason for the verbal advantage on start-unknowns. For both symbolic and verbal start-unknowns, students perform better using informal strategies (35% and 58%, respectively) than formal strategies (2% and 2%, respectively). However, students are more likely to use the formal strategy on symbolic start-unknowns (WI = 18%) than on verbal start-unknowns (WE = 4%). Thus, this greater use of the still fragile equation solving strategy on symbolic problems results in poorer performance. The EAPS model predicts that if students were to more frequently use informal strategies on symbolic start-unknowns, their performance would be substantially enhanced.

While strategy differences play a role in the relative difficulty of equations over verbal problems, it is differences in comprehension productions, not computational productions, that are the major source of the verbal advantage. Algebra equation solving is hard, at least initially, more because of the “reading of algebra” rather than the “doing of algebra.”

In general EAPS explains differences in problem difficulty in terms of differences in the productions involved (at Newell’s (1990) knowledge level) or differences in utility estimates (at Newell’s symbol level) or both. The verbal advantage explanation above illustrates how these two factors may combine to produce an explanation of relative problem difficulty. In some cases, explanations (or predictions) of problem difficulty derive simply from differences in productions involved. For instance, the effect that result-unknowns are easier than start-unknowns is explained in EAPS by the fact that solving result-unknowns involves a proper subset of the productions involved in solving start-unknowns. Although this effect is not surprising to most educators (Nathan & Koedinger, 2000), this explanation pattern is useful as it provides an example for generating explanations and predictions of similar results.

5.1.2 Explanations of Student Strategy Use

In addition to explanations of problem difficulty, the EAPS theory provides explanations for the nature and frequency of student strategy use. As we applied model development constraints, such as Choice Matching and Parsimony, the EAPS explanation of strategy use evolved. In EAPS1 we modeled student's informal solutions to start-unknown algebra problems as strategies that were explicitly selected and globally applied. Inspecting the resulting model, we noticed that the productions doing strategy selection were not playing any significant computational role. We found we could eliminate them without substantially changing the model's behavior. Instead of doing explicit global strategy selection and application, EAPS2 allows any production that applies to the current task demand to fire. Thus EAPS2 hypothesizes that students are not explicitly selecting these strategies but are simply behaving in reaction to local task demands. Students need not be aware of these strategies – they are a part of many students' implicit knowledge (Dienes & Perner, 1999).

5.1.3 Explanations of Student Errors

Besides insights into strategy use, our model development led to insights into the origin of student errors. A second weakness of EAPS1, besides explicit strategy selection, was a qualitative deviation from the data whereby it systematically under-predicted the frequency of students' conceptual errors on start-unknown problems. This deviation was the consequence of EAPS1 having explicit bugs (e.g., order of operations) as its only way of producing conceptual errors. In EAPS2, we added the possibility that the model might fail to find anything to do at any particular choice point. In this way, it produced conceptual errors not through explicit buggy knowledge but implicitly through the failure to have any productions with a sufficient estimated utility. By this account, while some student errors are systematic, a larger proportion of student errors are caused by a lack of knowledge. This explanation derived from the application of the Choice Matching Constraint.

We gained insight into the source of systematic errors by application of the Acquirability Constraint. In general in ACT-R, systematic errors are produced by production rules that yield incorrect results. In EAPS1&2, we simply wrote productions that produced the kinds of systematic errors we observed in student data. For instance, the EAPS2 production for order of operations essentially implemented the opposite of the order-of-operations rules – it added and subtracted before multiplying or dividing. In applying the Acquirability Constraint, we further pushed on the question of the source of systematic errors. How could a production like this one be learned? In ACT-R production rules are acquired by analogy to successful problem-solving episodes. While this does not require the resulting production will be guaranteed to be correct in all situations, it does necessitate that the production be correct in some situations. Thus, in EAPS3 we modified the order-of-operation production as well as other error producing productions, so that they are correct in some situations.

This change predicts that student performance on certain test items may over-estimate their true competence as student can get some types of problems correct for the wrong reason (c.f., Aleven, Koedinger, & Cross, 1999). EAPS3 (and theories like it) can predict the specific problem categories for which this over-estimation is possible and guide the design of assessments (and instructional activities) that better address true competence.

5.1.4 Explanations of Strategy and Error Frequencies

EAPS explains the frequency and stochastic nature of student strategies and errors through a combination of ACT-R production rules and production rule utility parameters. Different possible paths through the production rules yield different strategies and errors. The frequency of a particular strategy or error is determined by the number of paths that produce it and the probability of each path as computed from utility values of competing productions along that path. Any particular strategy or error frequency can be determined by following the production path or paths that lead to it and computing the corresponding probabilities as we have illustrated in the math model equations in Table 5 and Appendix 1.

5.2 Predictions: Using EAPS to Generate and Test Hypotheses about Learning

In addition to providing explanations of problem difficulty and student strategy and error behavior, the EAPS theory can be used for generating and evaluating hypotheses about student performance and learning. One class of predictions EAPS can make is about ordering the problems in a curriculum to better enhance learning. These predictions are based on three different features of EAPS: 1) production differences between problem categories, 2) utility differences between problem categories, 3) acquisition and competition of buggy versus correct productions.

5.2.1 Comprehension of equations translated from verbal problems:

One prediction EAPS makes has to do with comprehending equations. Our efforts to write transferable productions for EAPS3 led to a very similar underlying representation between verbal problems and equations. An implication of this commonality is the novel prediction that solving an equation a student produces from a word problem is easier than solving an equation they are given. When writing the productions, we noticed that parsing the first equation produced from a verbal problem is computationally redundant. The model has production rules for comprehending equations that create the verbal procedural representation, but when a verbal problem is translated into an equation the verbal procedural representation (the quantitative network) already exists. Therefore the production rules for comprehending equations do not need to fire.[10] This can be seen in the trace of equation solving on a story problem (Step 2 in Table 7), where EAPS3 moves immediately to focus on the last relation. In contrast to equation solving on story problems, for equation solving on symbolic problems EAPS3 must fire the equation comprehension productions to begin solving the problem (Table 8). Thus, a prediction of the model is that equation solving should be more successful in the context of doing a verbal problem than when it is done in isolation.

Post-hoc analysis of the DFA data are consistent with this prediction – even when we factor out possible selection biases in students or problems, for example, because it might be the better students who use formal algebra on verbal problems. Further research should test this prediction directly.

5.2.2 Predictions based on Production Utilities

ACT-R models, like EAPS, which have utilities attached to production rules, can make a general class of predictions about student learning and optimal instruction. One of these predictions is that instructional sequences should present problem categories that involve the use of a subset of productions before problems that use more. A related but more general prediction is before presenting a hard problem category identify the weakest component skill involved and present first a problem category that employs this skill and as few others as possible. The first of these instructional hypotheses is a common one (e.g., Gagne, 1985; VanLehn 1987) and predicts, for instance, that result-unknowns should be presented before start-unknowns. In fact, all textbooks we know of do present result-unknowns before start unknowns. The second instructional hypothesis is more subtle. Sometimes it is not possible to find a slightly easier problem category that has a subset of the productions involved in the target category. However, in such cases it may be possible to identify easier problem categories that contain a key difficult production in the target category, but the other productions involved are well practiced.

Consider the following example, which illustrates an instructional design strategy we call "bad-apple isolation". Our model predicts that start-unknown *word* problems should be presented before start-unknown *equations*. As can be seen in Table 5, verbal start-unknown problems require verbal comprehension (VC), operator and argument inversion (UC), and arithmetic (AP). Symbolic start-unknown problems require symbolic comprehension (SC), operator and argument inversion (UC) and arithmetic (AP). However, premature practice on symbolic problems could lead to a negative outcome. SC may fail to fire, or worse a buggy production (like OA) may fire. In the latter case, negative feedback will yield a reduction in utility for all productions that produced the result (Anderson & Lebiere, 1998, p. 255). Reducing the utility of buggy productions (UE) is appropriate, but other correct productions (like AP) will also be penalized if there is no way to assign blame to the real bug. We suspect that some cases of student motivation problems can be accounted for in this way, that is, by instruction beyond students' current preparation.

It is better to start with a representation the students are familiar with, the verbal representation. Because the utility of the verbal comprehension production (VC_2, 88%) is greater than that of symbol comprehension (SC_2, 61%) the new productions required to solve start-unknown word problems, Invert-

---

[10] Another possibility is that the symbolic comprehension productions do fire again, but only serve to strengthen the existing representation.

Operator and Invert-Argument (UC), can be strengthened independently of symbol comprehension (SC). This approach reduces the possibility of errors on more complex problems that build on them. After the student masters the Invert-Operator and Invert-Arguments productions, they can more easily go on to solve symbolic start-unknown problems.

Often multiple productions fire to produce an external action or solution that a student can get feedback on. If that feedback is negative, the learner has a difficult credit/blame assignment problem. Although the student's error may result because just one of these productions is bad, the brain does not know which one. Like other credit assignment algorithms (e.g., backpropagation (Rumelhart & McCllelland, 1986) or bucket brigade (Holland, Holyoak, Nisbett & Thagard, 1986)), in ACT-R some blame gets distributed to all knowledge elements involved. In such situations, "one bad apple spoils the whole bunch." In other words, correct productions that fire in a chain with an incorrect production will have their utility estimates decreased in such situations. If students are given premature practice on an overly difficult problem category, they may not only fail to learn. Worse yet, their performance may decline on easier problems as the utility of correct productions is reduced. To reduce the negative consequences of this "bad-apple effect" instruction should not only avoid such premature practice, but also provide prior practice that boosts the utility of productions that are later used in conjunction with difficult new skills.

In other words, in the example above prior practice on start-unknown problems helps boost Invert-Operator and Invert-Arguments (UC) and Arithmetic-Procedure (AP), so that when students later encounter start-unknown equations, credit/blame assignment can better isolate symbolic comprehension productions (SC). In some circumstances it is possible to identify simple problem categories that isolate the "bad apple" as the only the skill that needs to be applied. Often however, such problem categories cannot be created because skills, like symbol comprehension, do not produce any external behavior. In other words, it is not possible to create a problem where only symbol comprehension applies because that production does not yield any external behavior that can be checked.

5.2.3 Selecting and ordering instructional activities to reduce utility of overly-general productions

Another set of EAPS predictions is based on how EAPS models systematic errors as overgeneralizations. Like other learning mechanisms, ACT-R's analogy and utility learning mechanisms yield learning outcomes that are sensitive to the order in which problems are presented. In particular, some presentation orders can lead to a greater likelihood of over-general productions than others. In poorly designed instruction, the sequences of examples and practice problems may be such that students can initially learn and be successful with an overly general production. Continued practice on such problems will yield increasing utility estimates for the overly general productions. It will be then necessary to distinguish the overly general production from the correct one. This potential confusion suggests that curricula should have problems early on that will cause overly general rules to fail. In such cases, more specific correct productions increase in utility and with further practice will replace the over-general production.

An example problem that will cause an over-general production to fail is "800 – 40 x 4". Here, students often start by incorrectly subtracting 40 from 800. Problems like "800 – 40 x 4" should be introduced early so that overly general productions can receive negative feedback. Early error feedback can effectively eliminate such productions before they have a chance to accumulate a significant utility value.

Instructional prescriptions like this one may obvious, however, not only are they often violated in textbooks and classroom instruction, they are subtle because they seem in conflict with general (and worthwhile) prescription to start with easier problems and build to more difficult ones. Identifying the difference between the right kind of "easier" (e.g., as discussed in prior section on "bad apples") and the wrong kind of "easier" (as discussed above) is something that should not be left to intuition, but can be informed by detailed pedagogical domain theories like EAPS.

6. CONCLUSION

We have described a theory of early algebra problem solving, based on ACT-R, that predicts the kinds of correct and incorrect steps and the probability with which they are generated. The theory sheds light on the knowledge and knowledge selection processes behind student strategies and errors. It was developed not only based on ACT-R principles, but further guided by six model-building constraints that combine empirical data, mathematical modeling, and cognitive task analysis methods to create a pedagogical domain theory. This theory provides a powerful theoretical tool for educational psychologists and mathematics educators to both *productively generate* and *accurately evaluate* hypotheses about learning and instruction of quantitative reasoning and early algebra problem solving.

ACKNOWLEDGMENTS

This research was supported by a grant from the James S. McDonnell Foundation program in Cognitive Studies for Educational Practice, grant #95-11. Thanks to Vincent Aleven, Marsha Lovett, and Chris Schunn for their helpful comments and suggestions.

REFERENCES

Aleven, V., Koedinger, K. R., & Cross, K. (1999). Tutoring answer-explanation fosters learning with understanding. In Lajoie, S. P. & Vivet, M. (Eds.) Artificial Intelligence in Education, Open Learning Environments: New Computational Technologies to Support Learning, Exploration, and Collaboration, Proceedings of AIED-99, (pp. 199-206). Amsterdam: IOS Press.

Alibali, M. W. & Koedinger, K. R. (1999). The developmental progression from implicit to explicit knowledge: A computational approach. (Commentary on Z. Dienes & J. Perner, A theory of implicit and explicit knowledge.) Behavioral and Brain Sciences, 10, 755-756.

Anderson, J. R. (1993). *Rules of the Mind*. Hillsdale, NJ: Erlbaum.

Anderson, J. R. (1990). *The Adaptive Character of Thought*. Hillsdale, NJ: Erlbaum.

Anderson, J.R., & Lebiere, C. (1998). The Atomic Components of Thought. Laurence Erlbaum Associates.

Baranes, R., Perry, M., & Stigler, J. W. (1989). Activation of real-world knowledge in the solution of word problems. *Cognition and Instruction, 6*, 287-318.

Brown, A. L. (1994). The advancement of learning. *Educational Researcher, 23, 8*, 4-12.

Card, S.K., Moran, T.P., & Newell, A (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Carpenter, T. P., & Fennema, E. (1992). Cognitively Guided Instruction: Building on the knowledge of students and teachers. In W Secada (Ed.), Curriculum reform: The case of mathematics in the United States. Special issue of the *International Journal of Educational Research,* (pp 457-470). Elmsford, N.Y.: Pergamon Press.

Carraher, T. N., Carraher, D. W., & Schlieman, A. D. (1987). Written and oral mathematics. *Journal for Research in Mathematics Education*, 18 83-97.

Carraher, T. N., & Schliemann, A. D. (1987). Culture, arithmetic and mathematical models. Unpublished manuscript, Universidade Federal de Pernambuco, Recife, Brazil.

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

Dienes, Z. & Perner, J. (1999). A theory of implicit and explicit knowledge. *Behavioral and Brain Sciences, 10,* 755-756.

Gagne, R. M. (1985). *The Conditions of Learning and Theory of Instruction* (fourth edition). New York: Holt, Rinehart, and Winston.

Hall, R., Kibler, D., Wenger, E., & Truxaw, C. (1989). Exploring the episodic structure of algebra story problem-solving. *Cognition and Instruction, 6*, 223-283.

Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. R. (1986). *Induction: Processes of Inference, Learning, and Discovery*. Cambridge, MA: The MIT Press.

Kaput, J. J. (1999). Teaching and learning a new algebra. In E. Fennema and T. A. Romberg (Eds.) *Mathematics classrooms that promote understanding*. (pp. 133-155). Mahwah, N. J.: Erlbaum.

Kirshner, D. (1989). *The visual syntax of algebra*. Journal for Research in Mathematics Education, 20, 274-287.

Koedinger, K. R., & Anderson, J. R. (1993). Effective use of intelligent software in high school math classrooms. Artificial Intelligence in Education: Proceedings of the World Conference on AI in Education, AACE: Charlottsville, VA.

Koedinger, K. R., Anderson, J.R., Hadley, W.H., & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, *8,* 30-43.

Koedinger, K.R., & Tabachneck, H.J.M. (1994). *Two strategies are better than one: multiple strategy use in word problem-solving*. Presented at the annual meeting of the American Educational Research Association, New Orleans, LA.

Koedinger, K.R., & Tabachneck, H.J.M. (1995). *Verbal reasoning as a critical component in early algebra*. Paper presented at the 1995 annual meeting of the American Educational Research Association, San Francisco.

Koedinger, K. R. & Nathan, M. J. (submitted). The real story behind story problems: Effects of representations on quantitative reasoning. Manuscript submitted for publication.

Lewis, C. H. (1981). Skill in algebra. In J. R. Anderson (Ed.), *Cognitive Skills and Their Acquisition* (pp. 85-110).

Lovett, M. C., & Anderson, J. R. (1996). History of success and current context in problem-solving: Combined influences on operator selection. *Cognitive Psychology*, 31, 168-217.

MacLaren, B. A., & Koedinger, K. R. (1996). Toward a Dynamic Model of Early Algebra Acquisition. In Proceedings of the European Conference on AI in Education: Lisbon, Portugal.

Nathan, M. J., Koedinger, K.R., Tabachneck, H. T., Difficulty Factors in Arithmetic and Algebra: The Disparity of Teachers Beliefs and Students Performance. Paper prepared for The 1996 American Education Research Association Annual Meeting, New York.

Nathan, M. J., & Koedinger, K. R. (2000). *An investigation of teachers' beliefs of students' algebra development*. Cognition and Instruction, 18, 207-235.

Newell, A., (1990). Unified Theories of Cognition. Cambridge MA: Harvard University Press.

Newell, A., & Simon, H. A. (1972). *Human problem-solving*. Englewood Cliffs, NJ: Prentice-Hall.

O'Reilly, R. C. & Munakata, Y. (2000). *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*. Cambridge, MA: MIT Press.

Paige, J. M., & Simon, H. A. (1966). Cognitive processes in solving algebra word problems. In B. Kleinmuntz (Ed.), *Problem-solving* (pp. 51-119): John Wiley & Sons.

Polk, T. A., VanLehn, K. & Kalp, D. (1995). ASPM2: Progress toward the analysis of symbolic parameter models.

In P. Nichols, S. Chipman & R. L. Brennan (Eds.) Cognitively Diagnostic Assessment. Hillsdale, NJ: Erlbaum. pp. 127-139.

Ritter, F. E. (1992). A methodology and software environment for testing process models' sequential predictions with protocols. Doctoral Dissertation, Department of Psychology, Carnegie Mellon University.

Ritter, F. E., & Larkin, J. H. (1994). Developing process models as summaries of HCI action sequences. Human-Computer Interaction, 9, 345-383.

Rumelhart, D. E., & McCllelland, J. L. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition*. Cambridge, MA: Bradford Books.

Salvucci, D. D. (1999). Mapping eye movements to cognitive processes (Tech. Rep. No. CMU-CS-99-131). Doctoral Dissertation, Department of Computer Science, Carnegie Mellon University. (A thesis summary and the full thesis are available at http://www.cs.cmu.edu/~dario/TH99/)

Shalin, V., & Bee, N. V. (1985). Analysis of the semantic structure of a domain of word problems. (Tech. Rep. No. APS-20). Pittsburgh: University of Pittsburgh, Learning Research and Development Center.

Singley, M. K., & Anderson, J. R. (1989). *The Transfer of Cognitive Skill*. Cambridge, MA: Harvard University Press.

Tabachneck, H. (1992). Computational differences in mental representations: Effects of mode of data presentation on reasoning and understanding. Doctoral Dissertation. Carnegie Mellon.

Tabachneck, H. J. M., Koedinger, K. R., & Nathan, M. J. (1994). Toward a theoretical account of strategy use and sense-making in mathematics problem-solving. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

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*. (pp. 397-402). Hillsdale, NJ: Erlbaum.

Vygotsky, L. S. (1978). *Mind in Society*. Cambridge, MA: Harvard University Press.

VanLehn, K. (1987). Learning One Subprocedure per Lesson. *Artificial Intelligence* 31 (1987) 1-40.

VanLehn, K. (1990). *MindBugs: The origins of procedural misconceptions*. Cambridge, MA: MIT Press.

APPENDIX 1. COMPLETE MATH MODEL

To get a sense of the derivations of the equations in the math model of EAPS2 it is useful to work through some examples. For the simplest case, story integer result-unknowns, each quantitative relation is first comprehended (VC) and then arithmetic applies (AP), so the resulting equation for two quantitative relations is: VC * AP * VC * AP.

For a more complex example, formal algebra on a word integer start-unknown, there are two paths. In one path each relation is translated to an equation (WE), the equation is transformed symbolically (WI), comprehended (SC) and the arithmetic is performed (AP). In the second path the verbal problem is again translated into an equation, however, the equation is solved informally (because a student writes down an equation in this case, we coded this sequence in the data and model as a formal strategy). This path corresponds to translation (WE) followed by comprehending the equation (SC) and unwinding the relation (UC) and then performing the arithmetic (AP). These two possible paths are added together, resulting in the following expression: WE * WI * SC * AP * WI * SC * AP + WE * SC * UC * AP * SC * UC * AP.

Let us look at how probability values are calculated and propagated for the simplest case, Word Integer Result-unknown problems, which only use VC_1 and APe. VC competes only with the give-up production; APe competes with the give-up and the Arithmetic-Slip production (SL). The P and C values for VC, APe, SL and GU are, respectively, 6.6% and 3, 96.9% and 3, 86.8% and 3, and 25% and 0.[11]

First, we compute the expected gains for each production using PG–C. This leads to expected gain values for VC, AP, SL, and GU of 6.3, 16.4, 14.4 and 5 respectively. Next, we determine choice-point percentages for each production in each context in which it appears. Entering the expected gain values for VC and GU into the Likelihood Ratio equation yields the expression $e^{6.3/0.55}/(e^{6.3/0.55} + e^{5/0.55})$, which results in a value for VC_1 of 92%. Doing the same for AP_e and its competitors SL and GU yields 97%. Following the equation from Table 5 yields 80% correct performance on integer verbal result-unknown problems.

Below we show the complete math model for EAPS2, grouped by representation type and number difficulty (the six rows in Tables 1 & 2), with 11 equations for each (the eleven major columns across Tables 1 & 2). These 11 equations correspond with 4 solution codes for result-unknown problems (correct, arithmetic errors, conceptual errors, no answers), 3 for the start-unknowns solved informally (correct, arithmetic errors, conceptual errors), 3 for start-unknowns solved informally (correct, arithmetic errors, conceptual errors), and 1 more for no answer on start-unknowns.

A. Integer Story and Word Problems (separate rows in Tables 1 & 2 but they have the same equations)

1. $VC_1 * APe * VC_1 * APe$	**Correct Result-Unknown**
2. $VC_1 * (SLe * VC_1 * (APe + SLe) +$ $APe * VC_1 * SLe +$ $(APe + SLe) * VC_1 * (1 - APe - SLe))$	**Arithmetic Errors**
3. $VC_1 * (APe + SLe) * (1 - VC_1)$	**Conceptual Errors**
4. $(1 - VC_1) + VC_1 * (1 - APe - SLe)$	**No Answer**
5. $VC_2 * UC * APe * VC_1 * UC * APe$	**Correct Start-Unknown (Informal Strategy)**
6. $VC_2 * UC * (APe * VC_1 * UC * SLe +$ $SLe * VC_1 * UC * (APe + SLe) +$ $(APe + SLe) * VC_1 * UC * (1 - APe - SLe))$	**Arithmetic Errors**

[11] For the production costs we used 3 for most productions, 4 for the formal algebra, and 0 for the give-up production. The P values are determined by fitting the DFA data and the math model.

7. $VC_2 * UE +$ **Conceptual Errors**

$VC_2 * UC * (APe + SLe) * ((1 - VC_1) + VC_1 * (1 - UC))$

8. $WE * SC_2 * UC * APe * SC_1 * UC * APe +$ **Correct Start-Unknown (Formal Strategy)**

$WE * WI_2 * SC_2 * APe * WI_2 * SC_1 * APe$

9. $WE * SC_2 * UC * (APe * WE * UC * SLe +$ **Arithmetic Errors**

$SLe * SC_1 * UC * (APe + SLe) +$
$(1 - APe - SLe) +$
$(APe + SLe) * SC_1 * UC * (1 - APe - SLe)) +$

$WE * WI_2 * SC_2 * ((1 - APe - SLe) + (APe + SLe) * WI_1 * SC_1 * (1 - APe - SLe))$

10. $WE * (1 - SC_2 - OA - WI_2) +$ **Conceptual Errors**

$WE * SC_2 * (1 - UC - UE) + WE * SC_2 * UE +$

$WE * SC_2 * UC * (APe + SLe) * (1 - SC_1) +$

$WE * SC_2 * UC * (APe + SLe) * SC_1 * (1 - UC - UE) +$

$WE * WI_2 * (1 - SC_2) +$

$WE * WI_2 * SC_2 * (APe + SLe) * (1 - WI_1) +$

$WE * WI_2 * SC_2 * (APe + SLe) * WI_1 * (1 - SC_1) +$
$WE * OA$

11. $(1 - VC_2 - WE) +$ **No Answer**

$VC_2 * (1 - UC - UE) +$

$VC_2 * UC * (1 - APe - SLe)$

B. Integer Equation

1. $SC_1 * APe * SC_1 * APe$ **Correct Result-Unknown**
2. $SC_1 * (SLe * SC_1 * (APe + SLe) +$ **Arithmetic Errors**

$APe * SC_1 * SLe +$

$(APe + SLe) * SC_1 * (1 - APe - SLe))$

3. $SC_1 * (APe + SLe) * (1 - SC_1)$ **Conceptual Errors**
4. $(1 - SC_1) +$ **No Answer**

$SC_1 * (1 - APe - SLe))$

5. $SC_2 * UC * APe * SC_1 * UC * APe$ **Correct Start-Unknown (Informal)**
6. $SC_2 * UC * (APe * SC_1 * UC * SLe +$ **Arithmetic Errors**

$SLe * SC_1 * UC * (APe + SLe))$

7. $SC_2 * UE +$ **Conceptual Errors**

$SC_2 * UC * (APe + SLe) * ((1 - SC_1) + SC_1 * (1 - UC))$

8. $WI_2 * SC_2 * APe * WI_2 * SC_1 * APe$ **Correct Start-Unknown (Formal)**
9. $WI_2 * SC_2 * (APe * WI_2 * SC_1 * SLe +$ **Arithmetic Errors**

$SLe * WI_2 * SC_1 * (APe + SLe) +$
$(1 - APe - SLe) +$

$WI_2 * (APe + SLe) * WI_1 * SC_1 * (1 - APe - SLe))$

10. $WI_2 * (1 - SC_2) +$ **Conceptual Errors**

$WI_2 * (APe + SLe) * ((1 - WI_1) + WI_1 * (1 - SC_1)) + OA$

11. $(1 - SC_2 - WI_2 – OA)$ + **No Answer**
SC_2 * (1 - UC - UE) +
SC_2 * (UC + UE) * (1 - APe - SLe)

C. Decimal Story

1. VC_1 * (AS + APs) * VC_1 * (AS + APs) **Correct Result-Unknown**
2. VC_1 * ((SLs + BGs)*VC_1* ((AS + APs)+SLs + BGs) + **Arithmetic Errors**
   (AS + APs) * VC_1 * (SLs + BGs) +
   ((AS + APs)+SLs + BGs)*VC_1* (1 - (AS + APs)-SLs-BGs))
3. VC_1 * ((AS + APs) + SLs + BGs) * (1 - VC_1) **Conceptual Errors**
4. (1 - VC_1) + **No Answer**
   VC_1 * (1 - (AS + APs) - SLs - BGs)

5. VC_2* UC * (AS + APs) * VC_1 * UC * (AS + APs) **Correct Start-Unknown (Informal)**
6. VC_2* UC * ((AS + APs)* VC_1 * UC*(SLs + BGs) + **Arithmetic Errors**
   (SLs +BGs)*VC_1* UC * ((AS + APs) + SLs + BGs) +
   ((AS + APs) + SLs + BGs) * VC_1 * UC * (1 - (AS + APs) – SLs - BGs))
7. VC_2 * (1 - UC - UE) + VC_2* UE + **Conceptual Errors**
   VC_2* UC * ((AS + APs) + SLs + BGs) * ((1 - VC_1)+VC1*(1 - UC))
8. WE*SC_2*UC*(AS + APs) * SC_1* UC * (AS + APs) **Correct Start-Unknown (Formal)**
9. WE * WI_2 * SC_2 * (AS + APs) * WI_2 * SC_1 * (AS + APs) **Arithmetic Errors**

10. WE * SC_2 * UC * ((AS + APs) * SC_1 * UC * (SLs + BGs) + **Conceptual Errors**
    (SLs + BGs)* SC_1 *UC* ((AS + APs) + SLs + BGs) +
    (1 - (AS + APs) - SLs - BGs) +
    ((AS + APs) + SLs + BGs) * SC_1 * UC * (1 - (AS + APs) - SLs - BGs)) +
    WE * SC_2 * (1 - UC -UE) +
    WE * SC_2 * UE +
    WE * SC_2 * UC * ((AS + APs) + SLs + BGs) * (1 - SC_1) +
    WE * SC_2 * UC * ((AS + APs) + SLs + BGs) * SC_1 * (1 - UC - UE) +
    WE * WI_2 * ((1 - (AS + APs) - SLs - BGs) +
    ((AS + APs) + SLs + BGs) * WI_1 * SC_1 * (1 - (AS + APs) - SLs - BGs))
    WE * WI_2 * (1 - SC_2) + WE * WI_2 * ((AS + APs) + SLs + BGs) * (1 - WI_1) +
    WE * WI_2 * ((AS + APs) + SLs + BGs) * WI_1 * (1 - SC_1) +
    WE * OA +
    WE * (1 - SC_2 - OA - WI_2)
11. (1 - VC_2 - WE) + VC_2* (1 – UC) + **No Answer**
    VC_2* UC * (1 - (AS + APs) - SLs - BGs)

D. Decimal Word

1. VC_1 * APh * VC_1 * APh **Correct Result-Unknown**
2. VC_1 * ((SLh + BGh) * VC_1 * (APh + SLh + BGh) + **Arithmetic Errors**
   APh * VC_1 * (SLh + BGh) +

$(APh + SLh + BGh) * VC_1 * (1 - APh - SLh - BGh))$

3. $VC_1 * (APh + SLh + BGh) * (1 - VC_1)$ **Conceptual Errors**

4. $(1 - VC_1) +$ **No Answer**
   $VC_1 * (1 - APh - SLh - BGh)$

5. $VC_2 * UC * APh * VC1 * UC * APh$ **Correct Start-Unknown (Informal Strategy)**

6. $VC_2 * UC * (APh * VC1 * UC * (SLh + BGh) +$ **Arithmetic Errors**
   $(SLh + BGh) * VC_1 * UC * (APh + SLh + BGh) +$
   $(APh + SLh + BGh) * VC_1 * UC * (1 - APh - SLh - BGh))$

7. $VC_2 * UE +$ **Conceptual Errors**
   $VC_2 * UC * (APh + SLh + BGh) * ((1 - VC_1) + VC_1 * (1 - UC))$

8. $WE * SC_2 * UC * APh * SC_1 * UC * APh +$ **Correct Start-Unknown (Formal Strategy)**
   $WE * WI_2 * SC_2 * APh * WI_2 * SC_1 * APh$

9. $WE * SC_2 * UC * (APh * SC_1 * UC * (SLh + BGh) +$ **Arithmetic Errors**
   $(SLh + BGh) * SC_1 * UC * (APh + SLh + BGh) +$
   $(1 - APh - SLh - BGh) +$
   $(APh + SLh + BGh) * SC_1 * UC * (1 - APh - SLh - BGh)) +$
   $WE * WI_2 * SC_2 * ((1 - APh - SLh - BGh) +$
   $(APh + SLh + BGh) * WI_1 * SC_1 * (1 - APh - SLh - BGh))$

10. $WE * (1 - SC_2 - OA - WI_2) +$ **Conceptual Errors**
    $WE * SC_2 * (UE +$
    $(1 - UC - UE) +$
    $UC * (APh + SLh + BGh) * (1 - SC_1) +$
    $UC * (APh + SLh + BGh) * SC_1 * (1 - UC - UE)) +$
    $WE * WI_2 * (1 - SC_2) +$
    $WE * WI_2 * (APh + SLh + BGh) * ((1 - WI_1) + WI_1 * (1 - SC_1)) +$
    $WE * OA$

11. $(1 - VC_2 - WE) +$ **No Answer**
    $VC_2 * (1 - UC - UE) +$
    $VC_2 * UC * (1 - APh - SLh - BGh)$

E. Decimal Equation

1. $SC_1 * APh * SC_1 * APh$ **Correct Result-Unknown**
2. $SC_1 * ((SLh + BGh) * SC_1 * (APh + SLh + BGh) +$ **Arithmetic Errors**
   $APh * SC_1 * (SLh + BGh) +$
   $(APh + SLh + BGh) * SC_1 * (1 - APh - SLh - BGh))$
3. $SC_1 * (APh + SLh + BGh) * (1 - SC_1)$ **Conceptual Errors**
4. $(1 - SC_1) +$ **No Answer**
   $SC_1 * (1 - APh - SLh - BGh)$

5. $SC_2 * UC * APh * SC_1 * UC * APh$ **Correct Start-Unknown (Informal Strategy)**
6. $SC_2 * UC * (APh * SC_1 * UC * (SLh + BGh) +$ **Arithmetic Errors**
   $(SLh + BGh) * SC_1 * UC * (APh + SLh + BGh) +$
   $(APh + SLh + BGh) * SC_1 * UC * (1 - APh - SLh - BGh))$
7. $SC_2 * UE +$ **Conceptual Errors**
   $SC_2 * UC * (APh + SLh + BGh) * ((1 - SC_1) + SC_1 * (1 - UC))$
8. $WI_2 * SC_2 * APh * WI_2 * SC_1 * APh$ **Correct Start-Unknown (Formal Strategy)**
9. $WI_2 * SC_2 * (APh * WI_2 * SC_1 * (SLh + BGh) +$ **Arithmetic Errors**
   $(SLh + BGh) * WI_2 * SC_1 * (APh + SLh + BGh) + (1 - APh - SLh - BGh)) +$
   $WI_2 * (APh + SLh + BGh) * WI_1 * SC_1 * (1 - APh - SLh - BGh)$
10. $WI_2 * (1 - SC_2) +$ **Conceptual Errors**
    $WI_2 * (APh + SLh + BGh) * ((1 - WI_1) + WI_1 * (1 - SC_1)) + OA$
11. $(1 - SC_2 - WI_2 – OA) +$ **No Answer**
    $SC_2 * (1 - UC - UE) +$
    $SC_2 * UC * (1 - APh - SLh - BGh)$

APPENDIX 2. SAMPLE OF ACT-R PRODUCTIONS AND WORKING MEMORY ELEMENTS FOR EAPS3

For those readers interested in more details in the model, we illustrate the code of a number of key productions in EAPS3 and key elements of the working memory state before and after the application of those productions.

Comprehension of External Representation

Working Memory 1A shows the essential portion of working memory as EAPS3 begins solving "X * 25 + 10 = 110" in Trace 3 and Figure 2. As EAPS3 performs symbolic comprehension, productions fire to interpret the symbols in the equation one by one.[12] The productions that model this process remove terms from the list in the contents slot of the Equation WME and interprets them by creating corresponding *Quantity and *Relation working memory elements, shown in Working Memory 1B. The model's interpretation of the equation is stored in the interpretation slot of the Equation WME.

Working Memory 1: Equation working memory representation

A. Equation prior to comprehension

```
(Equation1    isa Equation
              contents (X * 25 + 10 = 110))
```

B. Equation and internal representations after partial comprehension[13]

```
(Equation1    isa Equation
              contents          ( =  110)
              interpretation  (Variable1  Op1  Number1  Op2  Number2) )
(Variable1 isa *Quantity   trans-to-alg          X    status Unknown )
(Op1          isa *Relation   op         Times )
(Number1      isa *Quantity   value      25             status Known  )
(Op2          isa *Relation   op         Plus )
(Number2      isa *Quantity   value      10             status Known  )
```

EAPS3 models correct equation comprehension using Comprehend-Quantity-Operator-Quantity productions, which pay attention to operator precedence, and build quantitative relations with the correct structure, instead of directly creating an arithmetic goal.

As the process of symbolic comprehension takes place, terms are removed from the "contents" list, interpreted (i.e., converted into *Quantity WMEs by the productions Comprehend-Variable, Comprehend-Operator, and Comprehend-Number) and placed in the "interpretation" slot. Normally, productions like Comprehend-Quantity-Operator-Quantity take a series of a *Quantity, *Relation, and *Quantity in the "interpretation" slot and build a node in the quantitative network, replacing the series in the "interpretation" with a *Quantity created to represent the result of that operation being performed.

---

[12] For example, Comprehend-Number, Comprehend-Variable, Comprehend-Operator. These productions are shown firing in Trace 4 in Appendix 1.

[13] This representation corresponds to the boxes in Figure 4, but the arrows connecting the quantities haven't been created yet.

Dealing with Unknown Input Quantities

Once the quantitative network has been built up, EAPS3 needs to focus on a quantitative relation to process. Production 1 shows one of the focusing productions, which simply marks the top-level Problem-Goal with the current focus of attention.

Production 1: Focus-on-Last-Relation Production

```
(p  Focus-on-Last-Relation
      =Goal> isa Problem-Goal
            focus NIL
            context =Equation
      =REL>   isa *Relation
            equation =Equation
            arg2      =Arg2
            result    =Res3
      =Arg2> isa *Quantity status Known
      =Res3> isa *Quantity status Known
==>
      =Goal>  focus =REL)
```

Once a quantitative relation has been focussed on, FORM-ARITHMETIC-GOAL-FOR-VERBAL begins processing by creating an arithmetic-goal, shown in Working Memory 2. It is basically a node in the relation network with the structure flattened. It contains the information in a quantitative relation, along with the information of the quantities contained in that relation.

Working Memory 2: Arithmetic Goal

```
(AG-1 isa Arithmetic-Goal
      arg1      Base-Wage
      op        Plus
      arg2      Tips
      arg2-status       Known
      arg2-value        66
      arg2-units        Dollars
      result   Gross-Salary
      result-status     Known
      result-value      81.90
      result-units      Dollars)
```

This arithmetic goal cannot be solved directly by straight arithmetic, because only the second argument value and the result value are known. There are two strategies for dealing with an unknown input quantity, Unwind (Table 9) and Interpretive Equation Solving (Table 10).

Productions that Implement the Unwind Strategy

Unwinding converts this arithmetic goal into one that can be solved using direct arithmetic. The first step in the Unwind strategy is to create goals to trigger the productions that implement it. This step is implemented by Set-Goals-to-Unwind, shown in Production 2 below.

Production 2: Set-Goals-To-Unwind Production

IF you have an arithmetic goal,
and you know what the second input and result are but not the first,
THEN create goals to invert the operator and invert the inputs and output and push them on the stack.

```
(p  Set-Goals-to-Unwind
      =goal>  isa Arithmetic-Goal
            object-status Unknown
            op              =Op
```

```
        object2        =Obj2
        object2-status Known
        result         =Res
        result-status  Known
==>
    =Inv-Op> isa Invert-Operator
        op     =Op
        new-op =Op
        new-op =New-Op
    =Inv-Args> isa Invert-Arguments
        object  =Res
        op      =New-Op
        parent =goal
    !push! =Inv-Args
    !push! =Inv-Op)
```

The goals Set-Goals-to-Unwind creates, goals to invert the arguments and the operator used in the relation, are shown in Working Memory 3 below.

Working Memory 3: Goals that trigger Invert-Arguments and Invert-Operator

```
    Inv-Args-1>
        isa           Invert-Arguments
        object            Base-Wage
        object2           tips
        object2-value     66
        result            Total-Wage
        result-value      81.9
        parent            pgoal-1
    Inv-Op-1>
        isa          Invert-Operator
        op            plus
```

After these two goals are created, the productions Invert-Input-Output and Invert-Operator, shown in Production 3, fire.[14]

Production 3: Productions that implement Unwind strategy

A. Invert-Input-Output

IF you have an arithmetic goal,
  and you know what the output and second input are but not the first input,
THEN combine the output and second input using the operator inverted to get the first input.

```
(p  Invert-Input-Output
    =Goal>     isa Invert-Arguments
        object     =Term1
        object2   =Term2
        result     =Term3
        parent    =Arithmetic-Goal
    =Term1> isa *Quantity status Unknown
    =Term2> isa *Quantity status Known
    =Term3> isa *Quantity status Known
    =Arithmetic-Goal> isa Arithmetic-Goal
```

[14] When a goal is created to invert an operator there are two slots, one for the old operator and one for the new. The slot for the inverted operator is initialized as the old one so if the goal is popped the operator remains the same, resulting in an unwind error.

```
==>
      =Arithmetic-Goal>
            object   =Term3
            object2 =Term2
            result    =Term1
      !pop!)
```

B. Invert-Operator

IF you have a goal to invert an operator,
  and you know what the inverse of the operator is,
THEN invert the operator and pop the goal.

```
 (p  Invert-Operator
   =Goal>  isa  Invert-Operator
      op         =Plus
      new-op   =Plus
   =Plus> isa  Op
      inverse  =Minus
      natural  =Plus
==>
   !output! ("So ~S becomes ~S" =Plus =Minus)
   =Goal>  new-op      =Minus
   !pop!)
```

After Invert-Input-Output and Invert-Operator fire and their associated goals are popped, the Arithmetic-Goal has been converted into one, shown in Working Memory 4, which can be solved using the standard arithmetic production, shown in Production 4.

Working Memory 4: Arithmetic Goal after Unwind

```
(AG-1 isa Arithmetic-Goal
     object              base-wage
     object-value        15.9
     op                  minus
     object2             tips
     object2-value       66
     result              total-wage
     result-value        NIL
```

Production 4: The Arithmetic Procedure

```
(p  Arithmetic
   =Goal>  isa Arithmetic-Goal
      object    =Obj
      op        =Nat
      object2   =Obj2
      result    =Res

=Obj>   isa *Quantity value   =ObjV        status Known
=Obj2> isa *Quantity value    =Obj2V       status Known
=Res>   isa *Quantity value   =ResV        - status Known

=Opr>   isa Op
   natural =Nat
   op =Op

!bind! =Result (!eval! (Perform-Arith-Op =ObjV =Op =Obj2V))
==>
 =Goal>
```

```
    result-value =Result
!pop!)
```

Main Production in the Interpretive Equation Solving Strategy

The second strategy for dealing with an unknown input quantity is formal algebra. The most reliable means for implementing this strategy was shown in Table 10. The main production that implements in is shown in Production 5.

Production 5: Write-Inverse-On-Both-Sides

IF you are focusing on the last operation in a side of an equation
  and that operation involves a known number,
THEN set goals to invert that operation
  And to write a new equation, with the inverse of that operation on both sides.

```
(p  Write-Inverse-On-Both-Sides
   =PG> isa Problem-Goal
        focus      =Operation
        context    =Equation
   =Operation> isa *Relation
        sem        =Op
        arg2       =Known-Number
        result     =RHS
   =Equation> isa  Equation
        contents-read  ($LHS  =Operation =Known-Number =Equal-Sign  $RHS)
    =Equal-Sign> isa *Object   sem          Equals
   =Known-Number>       isa *Quantity   status Known

==>
=Write-Eq>        isa Write-Equation
        parent     =PG
        old        =Equation
        add-num =Known-Number
        add-op     =Natural-Op
=Inv-Op>  isa  Invert-Operator
         op         =Natural-Op
        new-op     =Natural-Op
  !push! =Write-Eq
  !push! =Inv-Op)
```

When it fires, the two goals created: one that inverts the operator using the Invert-Operator production shown in Production 3, and one that writes a new equation concatenating an operator and number to each side. The processing of this equation then starts from the beginning.

Visual Equation Solving and Order-of-Operations Errors

Finally, let us look at the production that implements the order-of-operations error in Production 6. Because we view these errors being caused in part by Visual Equation Solving, it operates during parsing on the external Equation WME rather than on the quantitative network.

Production 6: Operate-on-Adjacent-Numbers

IF there is a problem goal with an equation,
and the equation has two numbers connected with an operator
THEN create an arithmetic goal to perform that operation and push it on the stack.

```
(p  Operate-on-Adjacent-Numbers
     =Goal> isa Problem-Goal
        context =Equation
     =Equation>              isa Equation problem =Prob
         interpretation      ($LHS1 =Number1 =Op =Number2)
     =Number1> isa *Quantity status Known
     =Number2> isa *Quantity status Known
     =Op>                isa *Relation   op    =Op-op
==>
     =RESULT> isa *Quantity status Unknown
     =Equation>
         interpretation      ($LHS1 =RESULT)
     =OO-Arith> isa Arithmetic-Goal
         object      =Number1
         op          =Op-op
         object2     =Number2
         result      =RESULT
     !push! =OO-Arith)
```