Tutoring Algebra Word Problems

Mitchell J. Nathan, Walter Kintsch,
& Clayton Lewis

Institute of Cognitive Science
University of Colorado, Boulder

Technical Report 88-12

Tutoring Algebra Word Problems

Mitchell J. Nathan, Walter Kintsch, & Clayton Lewis

Institute of Cognitive Science

University of Colorado, Boulder

Running head: TUTORING ALGEBRA

Abstract

Intelligent tutors for constructing proofs in geometry and solving algebraic equations are possible because these are closed, formal domains. Solving word problems is, in contrast, not closed since the student must employ common sense reasoning and infer information which is not stated in the problem in order to understand the situation described in the problem statement. A model for algebra word problem comprehension is presented. We distinguish several aspects of the problem comprehension process: Construction of a propositional textbase; qualitative representation of the problem situation; organization of the textbase into a formal conceptual model organized by problem schemata; and finally calculation of the solution from the problem model. A tutoring environment is described which embodies these principles. With it students can construct a graphical problem model which drives a computer animation of the situation. Students can compare the resulting animation to their internal situation model in order to evaluate and alter the problem model.

Tutoring systems need to be based both on an understanding of the psychological processes that students employ to work effectively in a domain, and on a didactic theory that specifies how these should be taught. In the present paper we present a theory of algebra word problem comprehension and solving -- the encoding strategies involved, the nature of the mental representations, and the calculation operations -- which provides a theoretical foundation for the development of a tutorial system.

A first sketch of this model has been presented in Weaver & Kintsch (submitted). It is an extension of the model for arithmetic word problem solving proposed by Kintsch & Greeno (1985). The latter model has been formalized and tested empirically, so that we have some assurance that it captures correctly important features of how first-grade students solve simple arithmetic word problems. It does not at present seem possible to formalize and test empirically the model of Weaver and Kintsch in the same way. In principle, the model for algebra can be specified as precisely as the model for arithmetic. However, in order to construct a simulation of that algebra model one would need a knowledge base that included an enormous amount of general world knowledge, in addition to knowledge about algebra. It is not known at present how to construct such a knowledge base, nor how to operate with it if we had one. Hence a simulation model is impractical at this time and we have, therefore, no ready means to derive and test empirical predictions from the model.

Instead, we shall test the model by building a tutor that assumes the empirical adequacy of the model. We follow Anderson, Boyle, &

Yost (1985) in this respect. Obviously this is a risky strategy. If our tutor fails to improve student problem solving performance and comprehension, we don't know whether the underlying theory was wrong, or whether we made poor design decisions unrelated to the theory. Similarly, if it works, we have only a very indirect test of the theory. As far as testing the model is concerned, however, that seems to be the best we can do at the moment. The tutor, however, is not just a test of the model, but, if successful, would be of interest in itself.

To build a tutor, we need to address both the question of *what* to tutor and of *how*. The answer to the first question is obtained by taking our model seriously. Although Weaver & Kintsch (submitted) merely sketch a model, this model can be worked out in the same detail as the model for arithmetic word problems, making explicit all the steps involved. Many of these steps, of course, people do not need help with -- our students can read, for instance. What we need to do is find those components of word problem solving that students find difficult. We shall take our cue from the work of Cummins, Kintsch, Reusser, & Weimer (1988) which suggests that, at least in the case of word arithmetic problems, the crucial and difficult step is the translation from the text into a formal, conceptual structure. Our goal will be to help students with that translation process.[1] In the next section we present a brief overview of the theory upon which we base our proposed tutor.

The answer to our second question -- how to tutor -- cannot be based solely on the process model of how students solve word

problems. A host of questions arise concerning various aspects of tutoring and user interface principles that are outside the domain of our process model . Some of these issues will be discussed in Section 2. Finally, we shall present our design for an algebra tutor in Section 3.

A Model of Algebra Word Problem Solving. The model of arithmetic word problem solving proposed by Kintsch & Greeno (1985), which is an instantiation of the general theory of discourse processing of van Dijk & Kintsch (1983), distinguishes several aspects of word problem solving: The construction of a propositional textbase, representing the meaning of the problem statement; the organization of this textbase in terms of the situation described in the problem and the mapping of this situation model, driven by the arithmetic schemata, into a conceptual problem model; and finally the calculation of the solution from the problem model.

Processing algebra word problems can be viewed similarly. First, a propositional textbase is formed, just as with any other text. This textbase is organized into a (qualitative) situation model and mapped into an algebraic (quantitative) problem model. A set of algebraic schemata will be described in the following section which allows one to construct problem models from episodes which are commonly found in algebra word problems. We show how equations can be derived from the problem model via constraint propagation. We shall first illustrate this process with a single, worked-out example, and then present a general discussion of algebraic schemata.

Example: A Distance-Rate-Time Problem. Consider the following word problem:

> A plane leaves Denver and travels east at 200 miles per hour. Three hours later a second plane leaves on a parallel course and travels east at 250 miles per hour. How long will it take the second plane to overtake the first plane?

The following proposition list can be derived from this text (as in van Dijk & Kintsch, 1983):

P1 LEAVE[PLANE1, DENVER]
P2 RATE[P1,200mph]
P3 DIRECTION[P1,EAST]
P4 LATER[P1,P5,3HOURS]
P5 LEAVE[PLANE2]
P6 DIRECTION[P5,EAST]
P7 LOCATION[P5,P8]
P8 [PARALLEL[COURSE]
P9 RATE[P5,250mph]
P10 HOWLONG[P11]
P11 OVERTAKE[PLANE2,PLANE1]

The top-level macroproposition for this text would be "PLANE2 overtakes PLANE1", with the information about how fast they are going, when they are starting, etc., subordinate to it. The

corresponding situation model might involve a description of the two speeding planes along a parallel course, ending when the second plane just passes the first. All of this would be just as in understanding a story. We assume that students have tremendous experience with this task.

Generating a Problem Model. In an algebra word problem, however, this situation model must be mapped into a formal algebraic problem model. That is, the relevant algebraic schema must be determined, and the situational information must be used to arrive at a solvable problem model. In constructing the problem model we note first that P2 specifies a relation between a distance and a time, thus eliciting a DRT schema labelled PLANE1. This schema has slots for D, R, and T, as well as a slot called "Specifications": P2 is assigned to Slot R, while P1 and P3 go into the Specification slot. P9 initiates the construction of a second DRT schema for PLANE2, fills its R slot, and assigns P5-P8 to the role of Specifications. P4, because it ties together the specifications of the two schemata, is tagged as a Supporting Relation. P10 is assigned to the T slot in PLANE2, and P11 serves as another Supporting Relation. Thus, propositions in the textbase serve as cues for the construction of a problem model, which organizes these propositions in terms of an algebraic schema.

The distinction between understanding the situation in everyday terms, and understanding the problem structure in terms of an algebraic schema is derived from Reusser (1988). As will be seen below, this is central to our approach to tutoring: The construction of the algebraic problem model requires the most support, while the

students' familiarity with the everyday situations in which these problems are embedded serve as a source of the support. We do not propose a stage theory, however, in which situational understanding must precede the formation of the problem model. The kind of situational understanding that is required for word problems must from the very beginning be driven by the salient features of the problem. The algebraic schema provides a perspective that is useful for the formation of a situation model. Thus the nascent formalism actually helps the student to understand the situation and better comprehend the problem statement.

We have chosen a simple graphical form to represent the problem model. Each DRT schema will be shown as three ovals, corresponding to D, R, and T, respectively, in a vertical arrangement, connected by line segments labelled "=" and "*". For clarity in analyzing this example, the text propositions organized by these schemata are written next to the ovals and/or lines to which they have been assigned. Inside the ovals quantitative information derived from the text propositions is shown. Line segments are labelled in such a way that the lines and ovals can be read as equations, either horizontally or vertically. Figure 1 shows the resulting graph.

Insert Figure 1 about here

The Situation Model-Problem Model Link. Note that in the construction of this problem model an important inference was made: P11 says only that one train overtakes the other. In our situation model we have one slower plane leaving initially and, after some delay, a faster plane approaching the first plane from behind. One can infer from the situation that the distances travelled by the two planes must be equal at the point of "overtake". It is crucial for understanding the problem that this inference be made and ultimately incorporated into the problem model. We will show later that failure to include such situation-directed inferences leads the tutoring system which embodies this model to construct an inappropriate animation. Seeing animation which does not correspond to the subject's expectations will hopefully lead the subject to alter the problem model appropriately. In this example, seeing the fast plane overtake the slower one and continue across the screen indefinitely may suggest to the student that the incorporation of an additional inter-schema relation in the problem model will bring about the expected animation on the screen. The top horizontal arc of Figure 1 which equates the distances of the two planes represents such a relation.

Equation Generation. The empty ovals in Figure 1 can be filled in by a process of constraint propagation. For instance, we can use the equation constructed on the basis of the supporting relation that the second plane started 3 hours later to obtain "h+t" as the time for the first plane. The two vertical DRT schemata can supply (ie. propagate) entries for their respective Distance ovals. Applying the inferred

equality relation for the two distances yields the equation 200*(t+3) = 250*t.

Problem models, and equations, can be constructed in much the same way as in Figure 1 for most algebra word problems found in college or high-school textbooks. We need merely a list of the required algebraic schemata.

Algebraic Schemata. Mayer (1981) has compiled a list of 1097 story problems from standard textbooks, which he classified into eight families. The first four families, comprising the majority of these problems, are various kinds of Rate problems, while the others are Number problems (such as Age problems), Geometry (e.g. Area problems), Physics (e.g. Ohm's Law), and Statistics (e.g. Probability). The general rate schema is of the form

UNIT1 :: RATE-UNIT1-PER-UNIT2 :: UNIT 2

The differences among the four rate families are differences in the nature of the units:

	UNIT1	UNIT2
FAMILY 1	amount	time
FAMILY 2	cost	unit
FAMILY 3	portion	total cost
FAMILY 4	amount	amount

Number problems, on the other hand, generally do not have a schema: All relations are specified in the text. For Physics problems, each

physical law or formula corresponds to a schema in our analysis. Similarly for Geometry problems, we need a schema for each formula and theorem. Mayer's 8th class, Statistics problems, could be treated in the same way, but we shall disregard it here.

We are essentially dealing with one rate schema and its variants for the bulk of the algebra problems, plus an open set of schemata corresponding to the geometry and physics formulas used, and some problems where a schema provides little in the way of constraints on selection of a problem solving strategy.

In Appendix A, two representative problem types are shown for each family. The many subtypes within each category listed by Mayer (1981) differ from the examples analyzed here in the nature of the supporting relations given and/or the unknown value asked for. These variations are represented similarly.

Note that important inferences are required to set up the problem models for Current and Work problems. In both cases the values for the rates involved are not directly given in the problem, but must be inferred: The speed of the boat upstream is boatspeed-plus-current; Mary's work rate is 1/5, because she needs 5 hours to complete (100% of) the job on her own, and so on. It is quite possible that these inferences might require the construction of an explicit model. In the case of the Work problem, e.g., one might want to set up two more work-schemata, one for the case where Mary works alone and one for the case where Jane works alone, making the computations of the work rates explicit. Figure 2 shows such a two-stage problem model.

Insert Figure 2 about here

The point made by Figure 2 is important. Problem models can be represented more or less completely. For some purposes a step can be left to be inferred (which is what we have done generally in the examples analyzed in the Appendix, for the sake of simplicity), while for other purposes, e.g. for tutoring, a more explicit analysis might be required. These inferences are not simple to make for most novices and can be a major source of error. Knowing the rate at which a person works is vital when relating the duration of labor to the amount of work done. This is likewise true for rate of travel in distance and current problems. The schematic formalism that we have developed makes students look carefully at the structural similarities across problem types. Students must still make the hard inferences. It will be more apparent, however, that this information is absent from the problem model and necessary for a coherent situation to be acted out on the computer screen.

Turning to Family 2, Cost-per-Unit-Rate, we have chosen a Fixed Cost and Dry Mixture as two examples in the Appendix. In each case, supporting relations specified in the problems must be used to construct the problem model. Family 3, Portion-to-Total-Cost-Rate, introduces a new element: Iterative constructions. Thus, the interest from the first year becomes part of the base amount for the second

year, and so on. Family 4 problems, Amount-to-Amount-Rate, require fairly subtle inferences, such as that the amount remains the same as pressure and volume vary, or that the amount of acid, too, can sum when forming a wet mixture.

Families 5-7 make use of a somewhat different graphical representation. Superficially, a formula like Distance = Rate X Time looks the same as Age-of-Ann = 2 X Age-of-Son, or Area = Length X Width. However, Rate (say, km/h) is a relational term,[2] explicitly linking Distance (km) and Time (h); hence we have written it in between D and T. Relational terms play a special role in word problems (Mayer, 1982, Weaver & Kintsch, submitted). "Length", however, is not a relational term, and it makes no difference whether in the formula for the area of a rectangle Length or Width comes first, while the ordering of Rate and Time is significant conceptually (though of course not mathematically). This equality between the two factors in non-rate problems is indicated in the graphic representation of Families 5-7 by writing both at the same level.

Number problems (Family 5) lack a schema; instead they are constructed entirely from the relations stated in the text. Geometry problems (Family 6), in contrast, involve many different schemata -- area and circumference of a rectangle, in our examples. Both physics examples given here (Family 7) involve the fulcrum-schema. Of course, many other geometrical and physical formulas are used in word algebra problems.

The rate schema in its various versions, plus a few common physics and geometry formulas are, therefore, all we need for the

construction of problem models for word algebra problems. Instead of 1,000 problem types, we need to be concerned with only a limited set of algebraic schemata. Our goal will be to teach students these schemata, help them to use these schemata in the construction of explicit graphical problem models, and show them how to derive equations from these graphs. Before we discuss our suggestions for tutoring, some general issues about tutoring need to be considered. The theory of word problem solving is concerned merely with the psychological processes involved, but in order to construct a tutor many other decisions will have to be made which are outside the scope of the theory.

Design of an Unintelligent Tutoring System. With the growing accessibility of powerful personal computers, graphics capabilities, and expanded memory, many researchers are looking to Intelligent Tutoring Systems (ITSs) as a panacea for remedial problems in education. A number of very impressive tutoring systems have indeed been developed (Polson & Richardson, 1988). The merits of this approach are basically three-fold. First is the belief that computer systems can be developed which embody lesson plans and present them to the student in an ecological manner with unfailing patience. Second is the individualized attention and feedback that a tutorial program can provide a student. Lastly, is the willingness with which students perform otherwise dull or tedious tasks on the computer.

While we do not feel that we can currently develop an *intelligent* tutoring system for word algebra problems, one which actually understands such problems, we do not feel this is a severe drawback.

We take a very conservative view as to the advantages of intelligent tutoring. There are good reasons why more modest tutoring strategies need to be considered as alternatives to intelligent tutoring. While it is true that computers can present materials in a variety of forms, there are many unanswered questions regarding information presentation and aesthetics. Most successful systems are developed through the skill and insight of an experienced designer, through trial and error, with a weak underlying design philosophy, if any.

The lack of a principled way to design interfaces is in itself no reason to abandon the ITS approach. We are far more concerned with the difficulty of achieving the levels of intelligence and flexibility that are required for successful, intelligent tutoring. The ability of current systems to tailor their behavior to the responses of the student is extremely limited. An intelligent tutor must understand *why* a student makes an error, and react accordingly. This requires a reasonably complete task analysis for the subject matter at hand. At present, such task analyses are done by hand, ruling out all but the most constrained of topics.

Secondly, for the program to adaptively customize its behavior, the tutoring system must understand what the student is doing. We need a psychological process model of the behavior in question. This is often not available, or too sketchy to be of much use. Intelligent tutoring without the necessary intelligence is a risky business. If the system misclassifies a student's response or behaves inconsistently, it can easily confuse or discourage the student (Holland, Holyoak, Nisbett, & Thagard, 1986). For most tasks, including mathematics,

there is no single, unique way for solving a problem. It is insufficient to evaluate a student's answer. It is the method of achieving this answer that intelligent tutoring systems need to evaluate. A program that misclassifies minor errors (or typos!) as major conceptual errors, or, worse, lets an erroneous method go uncorrected because it led to a correct answer, may do more harm than good. In the absence of a principled method for task analyses, of psychological processing models for the behavior to be tutored, or of tool kits for developing tutoring programs, intelligent tutoring for many domains doesn't look very promising.

On the third point, we largely agree. Teachers report high motivation on the part of students when computers are used in school work. Yet the potential for frustration is a concern that every tutoring system designer must address. The program must be robust. It must not blow up because of some unexpected input. It must terminate gracefully. Reasonable means of communication must be devised: Programs have an inherently limited vocabulary, so that the student cannot always obtain needed assistance during a session. Designers of tutoring systems and educational software cannot wait for the solution to the natural language processing problem.

Robust, flexible tutoring systems which are expected to exhibit intelligence may best be regarded as a long term goal. Our goal is the more immediate development of a computer system which aids students in problem understanding and learning. We propose an approach which assigns to the computer program tasks that it is good at doing **today**. Among these are simple, time-varying graphics,

bookkeeping, and processing formally described relations (e.g. equations). Tasks which lie outside of today's technology -- natural language processing, creative thinking, induction -- are left for the student. There are other systems, such as EUCLID (Smolensky & Fox, 1987), a computer system for specifying argumentation, which share this design philosophy. In our system the computer program serves as a fancy chalkboard for mathematics and animation. The student reads a problem, derives a formal problem model, and obtains feedback from the animation as to its correctness. The intelligence is all the student's. The system is there merely to help organize problem information around selected schemata and allow the student see the situational correlate for the specified formal relations.

Animating Word Algebra Problems. ANIMATE is **AN** Interactive system for **M**apping **A**nimation to **TE**xt. It is a computer system which facilitates comprehension of a text or story problem by helping the student to construct both an animated situation model and an accompanying problem model. The situation model (Van Dijk & Kintsch, 1983) is intended to capture graphically the overall gist of the passage. It is the basis for understanding the actions contained in the text. The problem model (Reusser, 1988) is an instance of an algebraic schema and serves as a guide for the parsing of a word problem into a set of equations which can then be solved using formal methods.

The set of equations ultimately produced when working in this environment is similar to those equations obtained using conventional methods. The path to it, however, is different. ANIMATE guides the

student to the ultimate set of equations using the schematic representations discussed above. We expect that this intermediate step will facilitate the formalization of the passage. Furthermore, feedback is provided by way of the animation so that the student may decide upon the correctness of the problem model. Finally, the system supports the transition from the schematic problem model to conventional equations.

Background. TRIP (Gould & Finzer, 1981) is the most notable system which provides students with a similar capacity to construct computer animated descriptions of word problems. This system was designed to help students formulate the appropriate Rate equation (Family 1) from a problem statement. As with our system, TRIP focuses on the translation process, decomposing it into three steps: Mapping the language into a meaningful graphical representation; identifying the "key relationships" within this representation; and transferring these relationships into an equation for algebraic processing. The first phase is performed closely with the teacher who provides the criteria for correctness of the constructed (ie. selected) picture. System-embedded knowledge is used to perform the second phase. The program has explicit knowledge of each problem assigned -- the distances, rates, travel times. The third phase makes use of a "Guess Table" which presents a series of the student's over- and under-estimations made for the value of the unknown variable in the problem. The Guess Table serves as data for the student to derive the analytical expressions which describes the problem formally. The resulting equation is then solved without the aid of the program.

As will be seen, while our system may be regarded as an offspring of TRIP, there are several important differences. Most significantly is the strong underlying theoretical framework of text comprehension and situation model formulation which drives our work. No such foundation is given for the TRIP system. The theory suggests two important deviations in the tutor design. First, we allow the student to supply the correctness criteria for the animation. We next provide the student with greater flexibility in the construction of situation and problem models. Students may play with the animation *before* the description is complete. Further, there is no notion of a *correct* problem description, only an internally consistent one. The student can arbitrarily vary a problem so that it has different values, constraints and unknowns. TRIP will permit the student to include only values which are correct for *that* problem and the animation can be run only when the problem description is correct and complete. Varying a problem, facilitated by stored problem templates and the use of Smalltalk-76, must be performed by the teacher.

While statistically reliable results are not available on TRIP, some valuable qualitative findings have been reported. Even fearful math students enjoyed using the computer. The system would likely be even more valuable if students were given as much control as possible over the dynamic actions of the animation. Thus greater flexibility needs to be provided. The transfer of "non-mechanical actions" (e.g. labelling the Guess Table) transferred poorly to paper and pencil performance since the program did these actions for the students. The performance of fairly complex tasks was performed

adequately by novices when they themselves constructed the scenario and had use of a cooperative user interface.

ANIMATE. According to the account of word algebra problem solving given above, the conceptual problem model plays a central role. Normally, however, it is merely an implicit, intermediate mental structure in a long line of such structures that are being generated from the initial perceptual stages of processing to the eventual overt response. What we present here, for the purpose of tutoring, is to make this structure explicit by representing it graphically, so that the student gains a clearer understanding of the conceptual relations in a word problem. We have good reasons to believe -- by extrapolation from the work on word arithmetic problems -- that much of the difficulty in solving word problems is tied to comprehending and setting up correct internal models of the problem text.

ANIMATE does not know the correct problem model for a word problem. It knows only that which is contained in the formal structure built by the student. Forcing students to be explicit and providing them with a framework for representing their notions of a problem will, we expect, be helpful. Thus, an explicit decision must be made by the student to employ and construct a DRT schema, for example, and construct the necessary relations. If an unconnected graph is constructed, the representation is surely incorrect. If some numerical information in the problem text is not included in the graph, something important may have been omitted. The formal constraints

of the graphical representation by themselves help the student arrive at the right conceptualization of a word problem.

ANIMATE provides the student with the ability to check the correctness of the problem model that has been constructed. This is done by running the animation -- time varying computer graphics -- which represent the conceptual model depicted in the network. The student compares this formally specified cartoon against his/her internal situation model of the problem statement. If they are consistent, the student is likely to have successfully translated the problem into a set of formal (and solvable) relations. If the animation misbehaves, something is wrong in the specification. It is then the task of the student to reevaluate the model, decide what aspect of the structure is inadequate (e.g.. an incorrect relation) and alter it to produce the expected animation. ANIMATE is incapable of helping the student to assess how well the animation matches the student's own internal situation model. ANIMATE knows nothing of situation models or the specific problem being addressed. It only does tasks that are easy for the computer: Given a formal specification of a problem -- a graph as in the Appendix -- it solves the underlying equations and then acts out based on stored internal scripts for various problem types, what it understands the situation to be. The system is capable only of making salient to the student the situation which has been formally specified by the network structure.To the student, however, evaluating the match is straight forward since it involves knowledge with which the student is quite familiar: When a

plane will leave and which will leave first; the resulting grey level of a mixture of light and dark shades; and so on.

When the student has constructed a schematic representation of the conceptual problem model, and checked its correctness against the animation, the final task is to derive an equation from the network which will lead to a solution of the unknown quantity. ANIMATE supports this process by maintaining the constraints inherent in the graphic representation. This part of the tutor is modelled after Greeno, Brown, Foss, Shalin, Bee, Lewis, & Vitolo (1986). It uses the links and values provided as constraints on the permissible values (numbers or expressions) which may fill empty nodes in the problem representation. ANIMATE works like a spreadsheet, although it does not immediately display its results. The empty nodes must be filled in by the student, but ANIMATE knows the correct answer and does not accept anything else. Indeed, after two failures, the student can ask the system to display the correct result.

Once all nodes and arcs in the graph are filled in with globally consistent numerical values, operators or formulas, the student can select any of the equations in the graph (by following any of the vertical or horizontal lines) and proceed to solve it. Although teaching how to solve these equations is not our concern, the tutor must be able to help students in this final phase of the solution process, too. We do this by prohibiting wrong moves by the student.

The goals of the tutoring system can be clarified by considering them in terms of the theoretical model presented earlier. We let the student read the problem text, form a text base and a situation model

without help. The student then uses ANIMATE to develop an algebraic problem model (Reusser, 1988) by constructing and filling in an algebraic schema. Icon selection and graph labelling help the student to cement the important features of the problem episode. The animation presents qualitatively the relations and values expressed formally in the schema and serves as a means of checking that problem model against the student's understanding of the situation. It is not necessary that this sequence be strictly observed: An incomplete problem model may be used to drive the animation of an isolated portion of the situation model. The resulting animation serves as feedback for elaborating or modifying the problem model. This process can be performed until both the problem and situation models conform to the student's expectation.

An Example: The DRT Problem Revisited. Consider the problem analyzed in Figure 1. On the basis of the problem text, the student must first decide which algebraic schema to use. A schema menu is entered which lists the names of common algebraic schemata, organized in families, as in the Appendix. Since the present problem involves Time, the student is cued that Family 1 is probably relevant, and within this family, the Distance-Rate-Time schema must then be recognized as the correct one. [3]

Suppose the DRT schema is selected. Three vertical ovals connected by lines appear on the screen, labelled D, R, T, =, and *, respectively. The student is then asked to name this schema, e.g. First Plane. A second DRT schema is then selected, and labelled Second Plane. Values for the rates are filled in for both planes. D is specified

as "miles" and T as "hours". A variable "t" is created for the time travelled by the second plane, because that is what the question asks for.[4] At this point the student may decide to try out the animation and let each plane fly.

Next, the student must recognize from the problem text that there are two supporting relations specified: The distances will be equal, and there is an equation relating the two time values. Horizontal lines and another oval are obtained from a menu and added to the graph. The lines are labelled, and the extra oval is is filled with the value specified in the text. Finding these supporting relations is crucial for successful problem solving. Both the graph structure and the animation provide support in this respect. If the D (or T) cells in the graph are not connected, the student learns to look for a possible relation in the problem text. If both planes in the animation leave at the same time, the student is cued that something is missing in the graph. The equality of the distances travelled when one plane overtakes the other is perceptually given in the animation. Thus, while the tutor doesn't know what the student is to do, it provides the student with many more cues for the solution of the problem than the text alone.

Running the animation is a menu-based operation. ANIMATE solves the equations specified in the graph, and constructs an animation on the basis of its Travel:Equal-Distance script. In order to select the right script, it asks the student certain questions: how many objects travel? are they travelling parallel to one another, in opposite directions, or back and forth? Two icons on a parallel course

and a clock (a gauge for the unknown variable) then appear on the screen The first starts moving, and after the specified number of time units, the second moves, with a greater speed. When the second object overtakes the first, the animation stops since all of the constraints given in the network are now met. Figure 3 shows what the screen looks like at the beginning and end of the animation run.

Insert Figure 3 about here

If the student has not constructed a correct problem model there are a number of cases that can be distinguished.

1. The conceptual model is insufficient. The animation, when started, shows no apparent motion since the icons are not tied to a time varying value.

2. The student tries to enter contradictory information into the graph. ANIMATE checks for constraint satisfaction every time an additional piece of evidence is added to the graph. As soon as it detects a contradiction, it alerts the student (who then must determine the source of the problem).

3. The animation works, but does not do what it is supposed to do (e.g., the faster plane takes off first, followed by the slow plane). The situation model and animation are semantically mismatched.

ANIMATE can only alert the student to certain types of errors, those which are *syntactically* verifiable given the current structure of the network. In these instances ANIMATE merely lets the student know that an error has been made. It does not know the true source of the error. However, given the structure provided by the graph and the animation, it is our belief that the student can figure out where the error occurred. If not, this is the time to consult a teacher! (A tutoring system where the teacher would have to intervene on some limited percentage of the problems is still an effective teaching device).

Suppose the student has gotten the animation to work correctly. There is still an equation to be derived, and solved. This is done by filling in the empty cells in the graph by means of the vertical or horizontal equations. ANIMATE will only accept correct values, and display the correct value if a student makes more than two mistakes on the same cell. Once all cells are filled in, the student can pick a suitable equation from the graph, in this case the one on the top for the equal distances. ANIMATE now enters into its equation-mode, accepting only correct manipulations of the equations.[5] If the student cannot solve the equation, ANIMATE supplies the answer or shows how the equation could be solved.

Animation Scripts. Similar animations can be constructed for other rate problems (Families 1-4 in the Appendix). Work problems show a container of unit size being filled at a certain rate: in 5 time units when Mary works alone, 4 if Jane works alone; If they work together, the rectangle is divided vertically into two parts, Mary's portion of the job being proportionally smaller than Jane's, and the rectangle fills faster than when either works alone.

Family II problems, e.g. coin problems, show sets of objects, their size indicating unit value, with associated bar graphs indicating total cost. Pie graphs most easily represent the structure for Family III problems. Note that we are constructing animations, not just another form of more or less schematic representation: capital and first-year interest are seen to combine to form a slightly larger circle for next year's capital; the discount is actually taken away from the old price, etc. Family IV problems can be animated using similar principles. We have no immediate plans for animating the problems in Families V-VII. Exactly how many scripts, and versions thereof, will be needed to handle all rate problems, or precisely what questions the student must be asked for our system to construct the right animation, is unknown at present. The answer depends crucially on the characteristics of the the computer system used to implement the animations.

Implementation. The ANIMATE system currently described is a prototype system built in Pascal on an Apple Macintosh Plus[6] personal computer. The system is built on top of NoPumpG, a constraint based programming environment which facilitates the construction of

computer graphics and animation through the use of spreadsheet machinery (Lewis, 1987).

System Evaluation. ANIMATE is in its prototype stage and will soon be used by college students to test empirically the theoretical ideas expressed in this paper. The empirical assessment is however a long-term and intensive task. Before the system is tested experimentally one can evaluate it on the basis of current user-interface and tutoring system principles. Fischer & Rathke (1988) have outlined a number of features of user-centered computer systems against which the proposed system can be measured.

Provided students have learned the conceptual problem model (which has been shown by Weaver & Kintsch (submitted) to be quite natural), ANIMATE, we expect, will be easy to learn and will require little background computer training. Thus, the overhead for learning the system should be small relative to the profit derived from it. Since it is largely non-modal, it provides a great deal of flexibility. Students can find their own solution path and graph structures. Objects and relations are defined by the user, not the system.

Objects and concepts are directly manipulable. Computation is performed graphically on icons intended to convey the semantics of each processing step (Hutchins, Hollan, & Norman, 1986). Syntax and commands are absent; instead there are physical actions and menu selections. Complex calculations are reduced through the spreadsheet mechanism to value and constraint propagation, eliminating the need for the user to pay attention to low level calculations and dependencies.

Conclusions

As yet an empirical evaluation of the model of algebra word problem comprehension described here and the effectiveness of a tutor built on these principles is not available. For such an evaluation, the actual computer system must be extended beyond the prototype that exists now so that students can work with it safely. One can nevertheless ask a number of questions about the system under development at this point, as well as our proposal for unintelligent tutoring. Our approach deviates in certain ways from the design principles espoused by Anderson et. al. (1985)[7]. Why do we think our system will be effective nevertheless? If it is not, are there reasonable modifications that could bring the system more in line with Anderson's design principles?

For our concerns, the most relevant work on tutoring by Anderson and his colleagues is that on teaching proofs in geometry (Anderson, Boyle, Corbett, & Lewis, 1986) and solving algebraic equations (Anderson et. al., 1985). We do not want to describe or discuss this work here, but merely point to the ways in which Anderson's tutoring strategy differs most crucially from ours. Anderson's tutor understands what the student does. When the student deviates from the correct solution path, the system puts the student right back on track. Anderson shows that once students are lost, it is very hard to set them straight. His tutoring strategy is successful because it gives immediate feedback when errors are still easily recognized and corrected.

Geometric proofs and algebraic equations are closed, formal systems, and Anderson has shown that it is possible to construct a computer system that understands what is going on when students work such problems. Given this system, his tutoring strategy of immediate feedback becomes possible. Word problems , in contrast, are open systems, as are many other domains in which tutoring is of interest. We have repeatedly argued that it is not feasible today to build systems which are sufficiently intelligent for tutoring in these domains. In consequence, we cannot provide immediate feedback to the student solving a word algebra problem, nor correct him or her at every wrong step.

According to Anderson, our students will get hopelessly lost very quickly, thrashing about in a solution space which they do not understand. But there is a difference in this respect between doing geometry proofs and solving equations on the one hand and processing word problems: The former are formal, self-contained tasks, devoid of every-day meaning and complexity, while word problems are presented in informal language and often deal with familiar, well-understood contexts. It may be that the chunk-size of problems in meaningful contexts is much greater. Will our students, if they have constructed an incorrect problem model and an finally realize that the animation is not doing what it is suppose to do, be able to locate and correct their error? It is possible. If the faster train starts before the slower one, the student is not left with a puzzle, but knows what needs to be repaired; if the interest added to a capital investment makes the total amount shrink, the credit assignment

problem may not be too difficult (as is often *not* the case with geometric proofs). Word problems are hard because fuzzy, natural language text needs to be put into a precise, canonical form, for which vast amounts of ill-specified knowledge is required. At the same time, word problems have an advantage, because they involve concrete, familiar, easily understood situations. Thus, immediate feedback, and small steps may not play as much a role here as in Anderson's work.

Of course, it would not be too surprising if we would find out that some of the situations involved are not as well understood as one might suppose. What happens when a train overtakes another is surely quite clear to our students. But consider the discount problem presented in the Appendix. Students go shopping, too, and know about discounts -- that they are good things, and you should buy if the discount is large. Just how clear they are about the relation between the original price, sale price, and discount may be another matter. We may find that students have trouble with discount problems, unless we explicate for them that relationship as part of the problem statement. In that case we would know what to teach: What a discount really is! This would resemble what we learned about first-graders doing word arithmetic problems: What they need most is to be taught the meaning of words and phrases such as "have-altogether", or "have-more than".

Finally, suppose our tutor doesn't work. Students do get hopelessly tangled up in the construction of the problem model and animation. One possibility is that the tutor failed for the reasons suggested by Anderson: The lack of immediate feedback, and the consequent difficulty students have with credit assignment. There are

ways in which our tutor could be made more like an Anderson tutor, without adding artificial intelligence. A set of word algebra problems could be prepared for which the correct problem representation is stored in the system, so that the system would know what to expect, and could more actively help the student.

Of course, if the tutor fails, this may simply mean that our notions about how word algebra problems are processed are wrong. Our tutor is an exploration of the efficacy of a conceptual graphic representation intermediate to the text and the underlying equation. We propose this intermediate level because our theory assigns it an essential role in the process of word problem solving. Students may find working with this intermediate representation more of a bother than help. Studies of how experts solve physics problems (Larkin ref.) suggest otherwise. Even for the simplest problems, experts start with a schematic, graphic representation, while novices jump right into the equations. Even in physics, schematic problem representations are neither systematic nor explicitly taught. It is something that experts pick up in the course of becoming an expert. Larkin et al. teach graphic representation schemes to physics students. We expect a similar strategy to work well in teaching students word algebra problems.

The position that we have taken throughout this paper has been that students need to *understand* the situation in order to solve word problems. There is however, a syntactic alternative opposed to this semantic view: Students need to use the symbol crunching aspects of algebra, without regard to their ability to understand the situation.

What counts is that they solve the problem. After all, many real problems are too complex for situational understanding! Situational understanding is an undesirable crutch in this view. Kintsch (1988) has discussed cases where situationally rich arithmetic word problems are solved simply by exploiting the familiar situational constraints in these problems, with questionable profit for arithmetic training. Similar examples can undoubtedly be found in the word algebra domain. However, even if semantic strategies may not be the final goal for engineers and mathematicians, they may be a very good thing for the average student, and permit more of them to achieve higher levels of performance.

General Footnote:

This research was supported by NSF-Grant BNS-8608741 to Walter Kintsch. We also thank Peter Polson for many valuable discussions of this project, and Kurt Reusser and Denise Dellarosa Cummins, whose comments greatly helped us in writing this paper.

References

Anderson, J. R., Boyle, C. F., Corbett, A., & Lewis, M. (1986). *Cognitive modeling and intelligent tutoring* (CMU Tech. Rep.). Pittsburgh, PA: Carnegie-Mellon University, Psychology Department.

Anderson, J. R., Boyle, C. F., & Yost, G. (1985). The geometry tutor. In A. Joshi (Ed.), *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 1-7). Los Altos, CA: Morgan Kaufmann.

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

Fischer, G., & Rathke, C. (1988). Knowledge based spread sheets. To appear in *Proceedings of AAAI* (St. Paul).

Gould, L., & Finzer, W. (1981). A study of TRIP: A system for animating Time-Rate-Distance problems. *Proceedings of the 3rd World Conference on Computers in Education.* (Lausanne).

Greeno, J. G., Brown, J. S., Foss, C., Shalin, V., Bee, N. V., Lewis, M. W., & Vitolo, T. M. (1986). Cognitive principles of problem solving and instruction. Report 41, Berkeley Cognitive Science Report Series.

Hall, R., Kibler, D., Wenger, E., & Truxaw, C. (1986). Exploring the episodic structure of algebra story problem solving. Technical Report 86-24, Irvine Computational Intelligence Project.

Holland, J. H., Holyoak, K., Nisbett, R. E., & Thagard, P. R. (1986). *Induction: Processes of inference, learning, and discovery.* Cambridge, MA: MIT Press.

Hutchins, E. L., Hollan, J. D., & Norman, D. A. (1986). Direct Manipulation Interfaces. In Norman, D.A. and S.W. Draper (Ed.s), *User Centered System Design: New Perspectives on Human-Computer Interaction.* Hillsdale, NJ: L. Erlbaum.

Kintsch, W. (1988). The role of knowledge in discourse comprehension: A construction-integration model. *Psychological Review, 95,* 163-182.

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

Larkin, J. H., McDermott, J., Simon, D. P., & Simon, H. A. (1980). Models of competence in solving physics word problems. *Cognitive Science, 4,* 317-345.

Lewis, C. (1987). NoPumpG: Creating interactive graphics with spreadsheet machinery. Technical report CS-CU-372-87. Department of Computer Science, University of Colorado.

Mayer, R. E. (1981). Frequency norms and structural analysis of algebra story problems into families, categories, and templates. *International science. 10.* 135-175.

Mayer, R. E. (1982). Memory for algebra story problems. *Journal of Educational Psychology. 74.* 199-216.

Polson, M. C., & Richardson, J. J. (1988). *Foundations of Intelligent Tutoring Systems.* Hillsdale, NJ: L. Erlbaum.

Reusser, K. (1988). From text to situation to equation: Cognitive simulation of understanding and solving mathematical word problems. Research report no. 5. Universitat Bern, Switzerland.

Smolensky, P., & Fox, B. (1988). Computer-aided reasoned discourse. In R. Guidon (Ed.), *Cognitive Science and its Application to Computer Human-Interaction. Hillsdale,* NJ: L. Erlbaum.

van Dijk, T. A., & Kintsch, W. (1983). *Strategies of Discourse Comprehension.* New York: Academic Press.

Weaver, C. A., & Kintsch, W. (submitted). The conceptual structure of word algebra problems.

Appendix

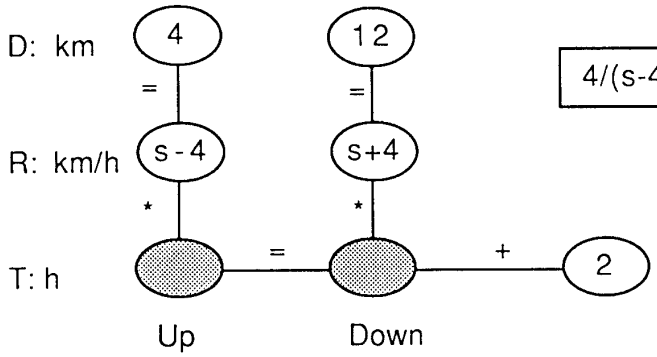
Appendix

Problem Models for Two Examples from each of Seven Problem Families, after Mayer (1981)

FAMILY I: AMOUNT-PER-TIME RATE

Current

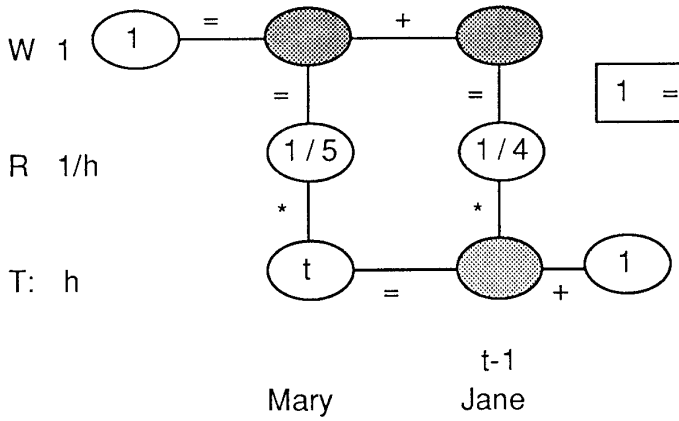
The current in a stream moves at a speed of 4 km/h. A boat travels 4 km upstream and 12 km downstream in a total time of 2 hours. What is the speed of the boat in still water?



$$4/(s-4)+12/(s+4) = 2$$

Work

Mary can do a job in 5 hours and Jane can do the job in 4 hours. If they work together, how long will they take to do the job if Jane starts 1 hour after Mary?

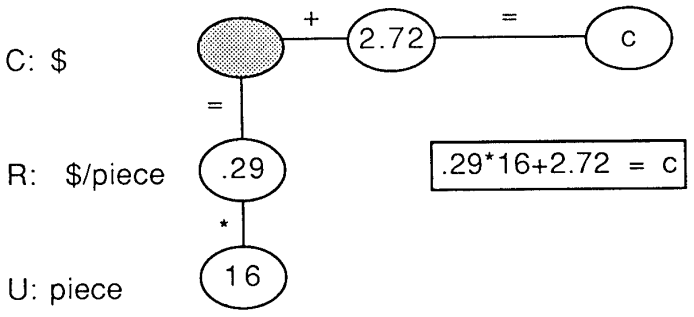


$$1 = 1/5*t+1/4*(t-1)$$

FAMILY II: COST-PER-UNIT RATE

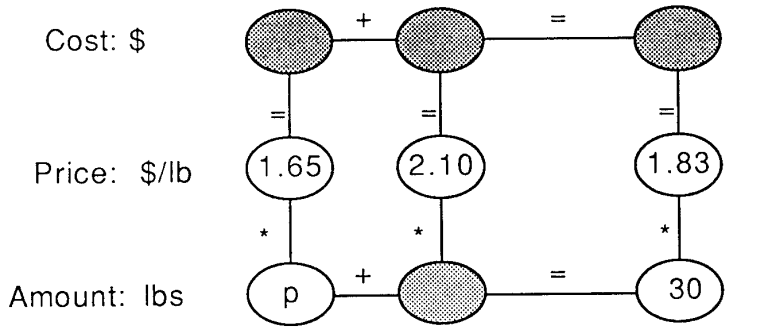
Fixed Cost

Sixteen balls of yarn can be bought from a mail order housefor 29c each plus $2.72 for postage. What does the total order cost?



Dry Mixture

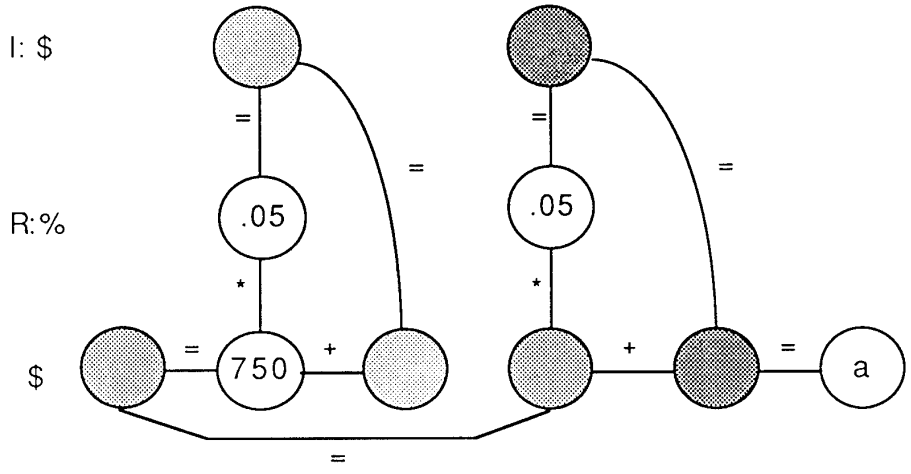
A grocer mixes peanuts worth $1.65 a pound and almonds worth $2.10 a pound. She wants 30 pounds of the mixture worth $1.83 a pound. How many pounds of each should the grocer include in the mixture?



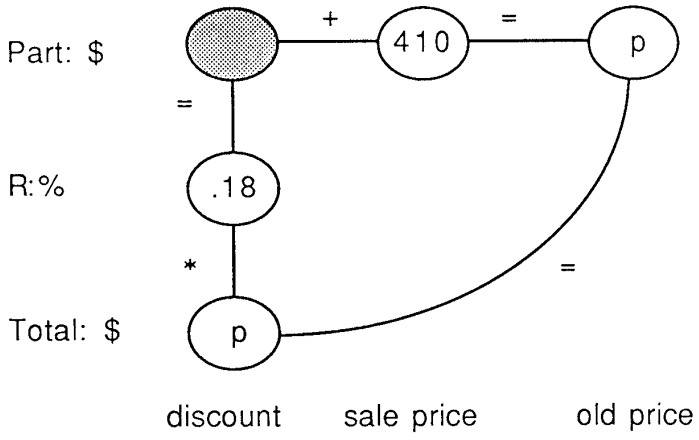
$$1.65p + 2.10(30-p) = 1.83*30$$

FAMILY III: PORTION-TO-TOTAL-COST RATE

Suppose $750 is invested at 5% annually. What amount will be in the account at the end of 2 years?



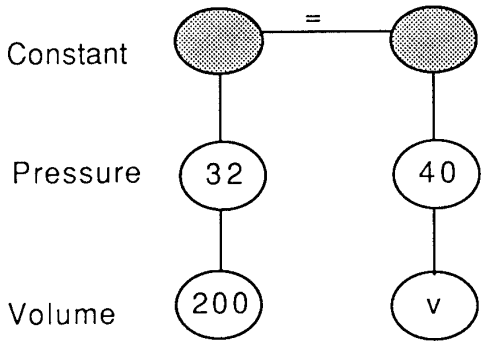
An appliance store drops the price of a certain TV 18% to a sale price of $410. What was the former price?



FAMILY IV: AMOUNT-TO-AMOUNT RATE

Inverse Variation :

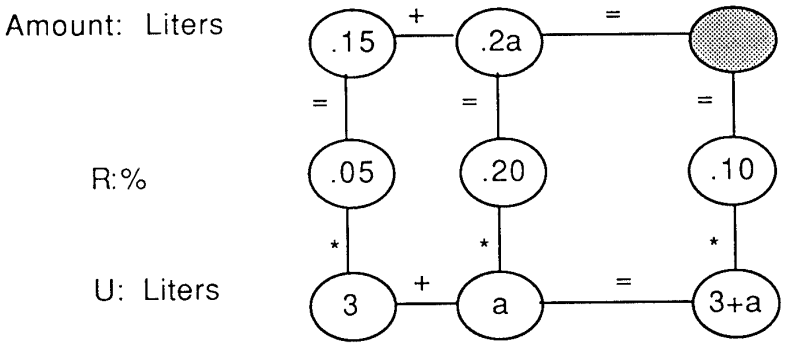
The voluime of gas varies inversely with the pressure on it. The volume of gas is 200 cc under a pressure of 32 kg/sqcm. What will be its volume under a pressure of 40 kg/sqcm ?



$$32*200 = 40v$$

Wet Mixture :

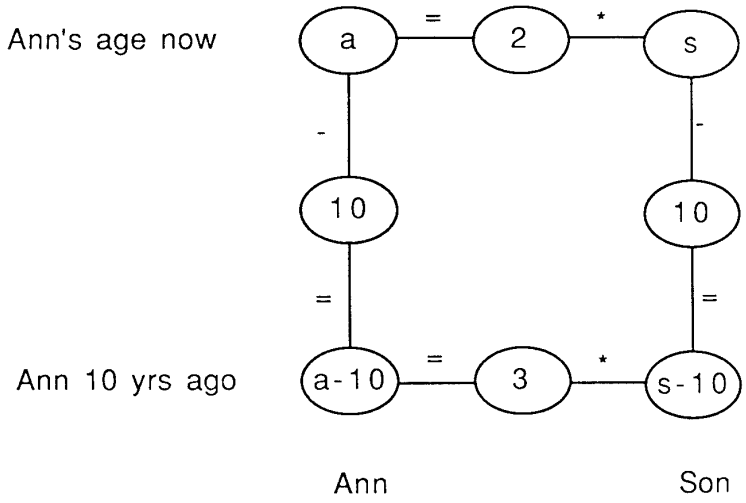
A chemist has 3 liters of a 5% acid solution. How many liters of a 20% acid solution must be added to make a mixture which is 10% acid?



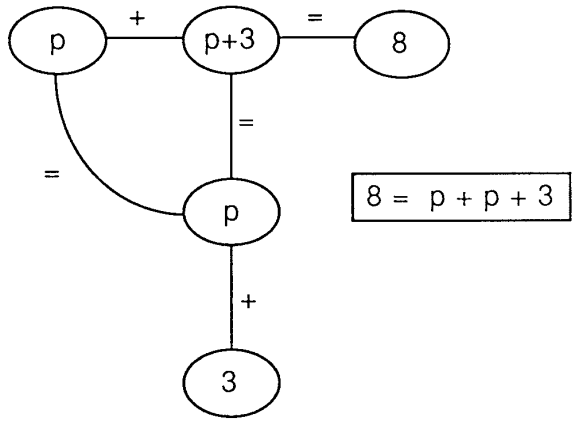
$$.05*3 + .20*a = .10*(3+a)$$

Family V: Number

Ann is twice as old as her son. Ten years ago, she was three times as old.
How old is she now?

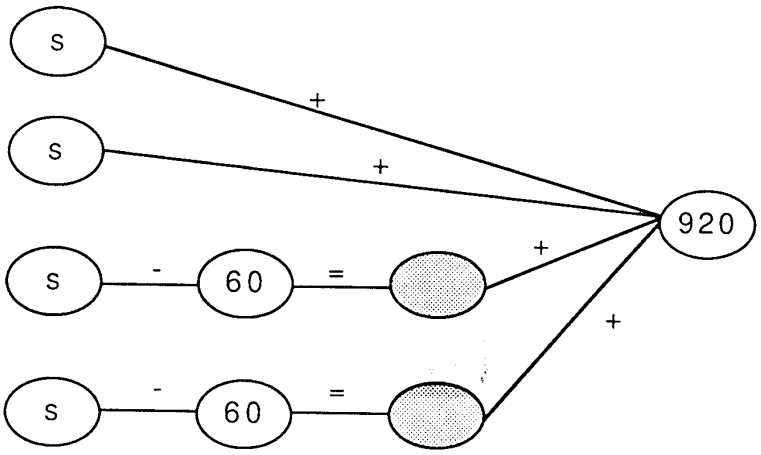


An 8 m rope is cut into two pieces. One piece is 3 m longer than the other.
How long are the pieces?



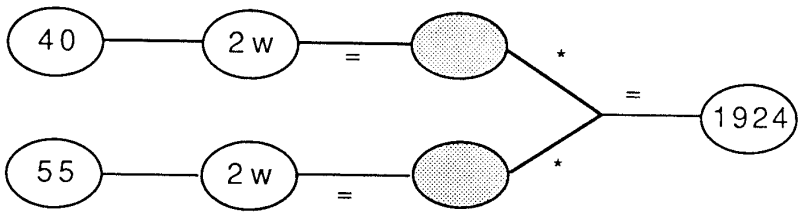
Family VI: Geometry

A rectangular playground is 60 m longer than it is wide. It can be enclosed by a 920 m fence. How long is it it?



$s + s + (s-60) + (s-60) = 920$

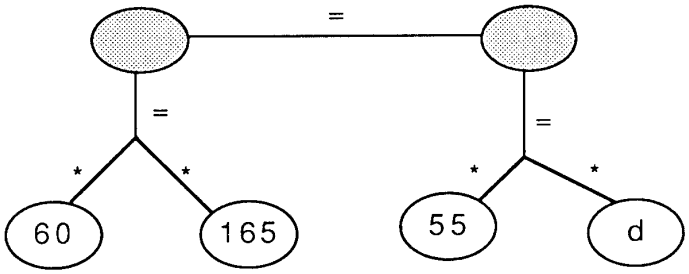
A framed mirror is 40 cm x 55 cm. 1924 sqcm show. Find the width of the frame.



$(40-2w)*(55-2w) = 1924$

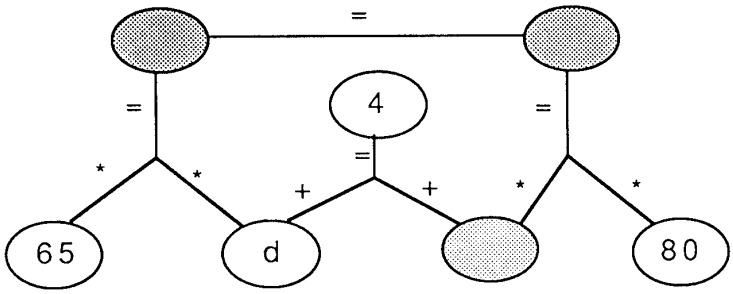
Family VII: Physics

Laurie weighs 60 kg and is sitting 165 cm from the fulcrum of a seesaw. Bill weighs 55 kg. How far from the fulcrum must Bill sit to balance the seesaw?



$$60*165 = 55*d$$

Tina and Wilt are sitting 4 m apart on a seesaw. Tina weighs 65 kg, and Wilt weighs 80 kg. How far from the fulcrum must Tina be sitting if the seesaw is in balance?



4 - d

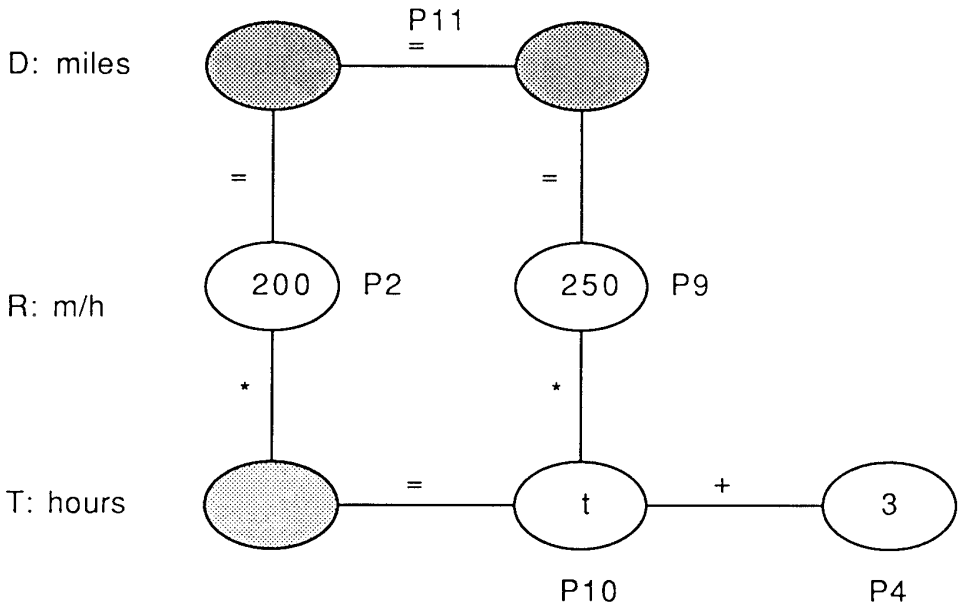
$$65*d = 80*(4-d)$$

Figure Captions

Figure 1. Graphical representation of the problem model for a typical DRT schema.

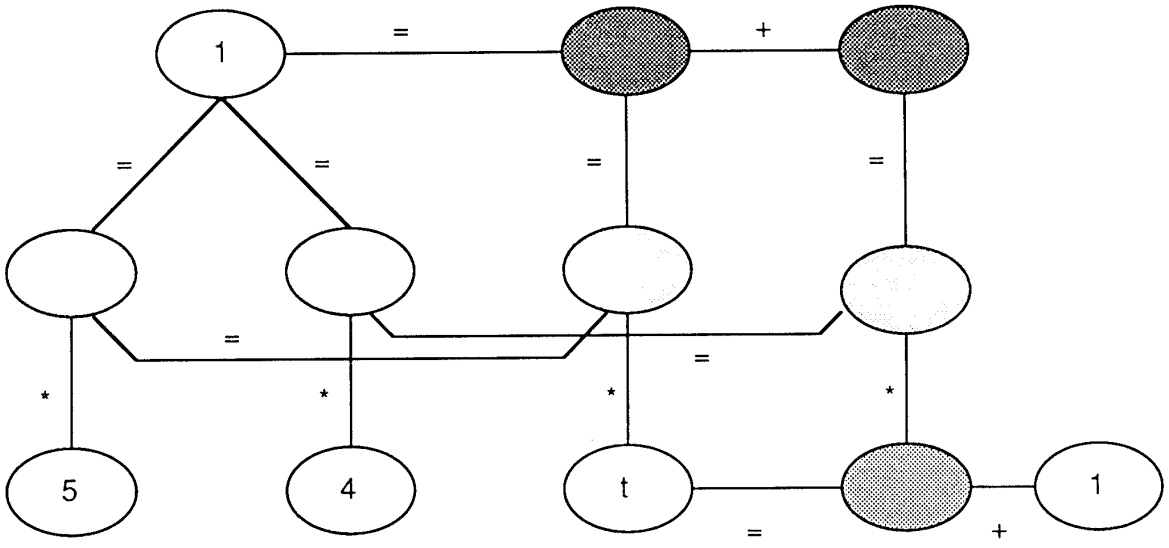
Figure 2. A two-stage problem model for a typical Work problem which makes certain relations and computations explicit.

Figure 3. (a) - (d) Successive screens for representing a typical Overtake problem using the prototype of the **ANIMATE** tutoring system. Insufficient specification of the problem model leads to an inappropriate situation model. (e) When a conceptual error is made, such as an incorrect sign for the delay, animation which does not match the subject's situation model is created.



PLANE 1
P1
P3

PLANE 2
P5
P6
P7
P8

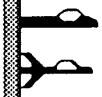


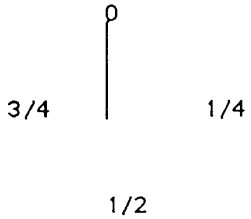
Mary alone Jane alone Mary Jane

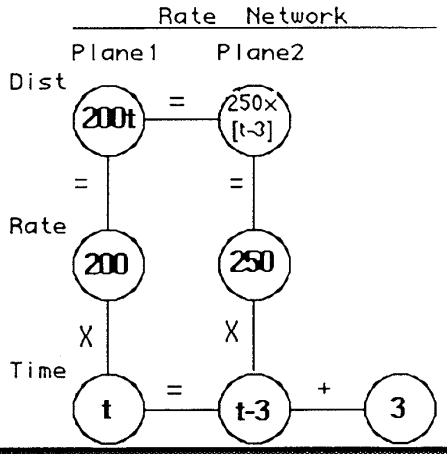
together

File Edit Make Show Ops Clock Network

ANIMATE







File Edit Make Show Ops Clock Network

ANIMATE

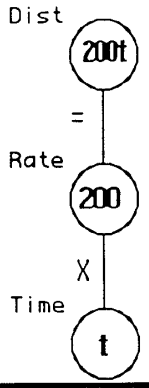
0

3/4 1/4

1/2

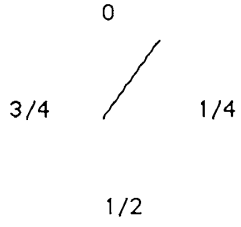
Rate Network

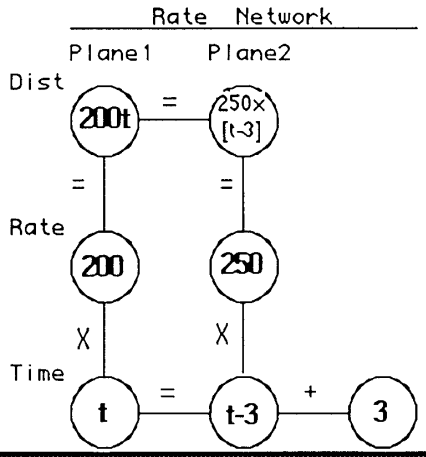
Plane1

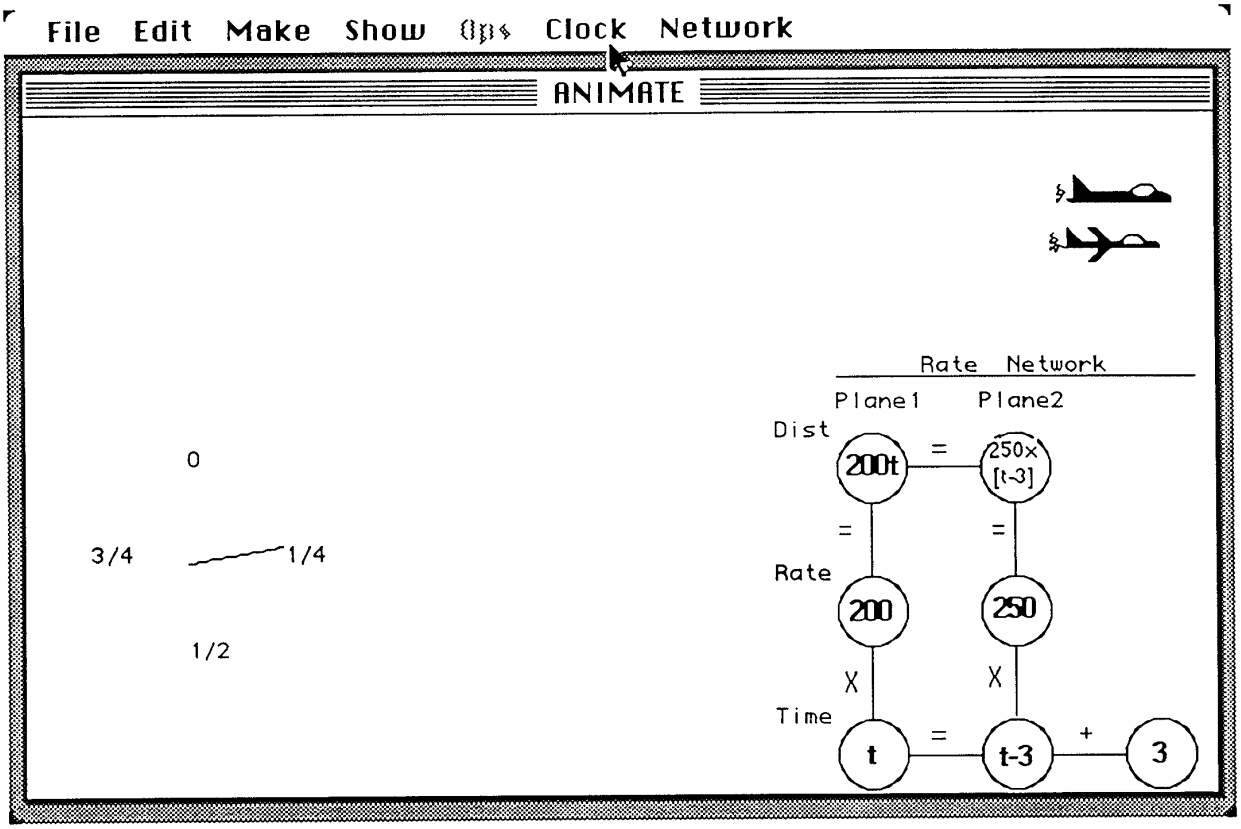


File Edit Make Show Ops Clock Network

ANIMATE

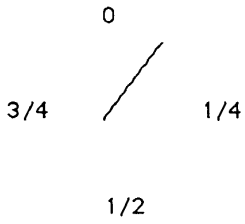


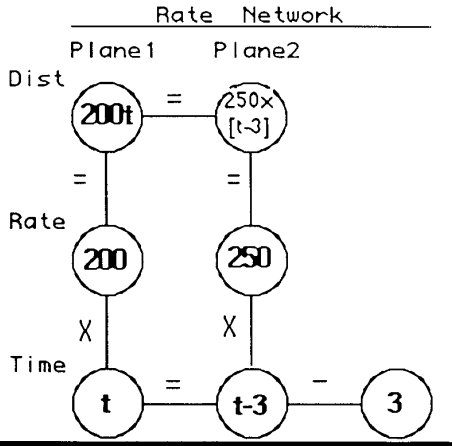



File Edit Make Show Ops Clock Network
ANIMATE
0
3/4
1/4
1/2
Rate Network
Plane1
Plane2
Dist
200t
=
250x
[t-3]
=
=
Rate
200
250
X
X
Time
t
=
t-3
+
3

File Edit Make Show Ops Clock Network

ANIMATE





Endnotes

[1] Students also need help in solving the algebraic equations. We shall be concerned with the construction of conceptually correct equations. Other tutors (e.g. Anderson, 1986) exist for solving the equations.

[2] "Relational terms" are quantities or variables which relate the behavior or value of two other variables. A "relational proposition" is a proposition (van Dijk & Kintsch, 1983) which contains a relational term as an argument.

[3] If the student is uncertain, more information about each schema can be requested, such as common measures (km/hr) or prototypical problems.

[4] Once the student has decided on t as a variable, the system will not allow a second variable: all other cells must be numbers or formulas involving t.

[5] This is not meant to teach students how to solve equations. For that purpose, a program such as Anderson et al (1986) is needed.

[6]The terms "Apple", "Macintosh" and Macintosh Plus" are trademarks of Apple Computer, Inc.

[7] We are using Anderson's work here as a concrete example of a successful intelligent tutoring system. There are, of course, other comparisons that could have been made.