Behavior Research Methods, Instruments, & Computers
1986, 18 (2), 147-154

A computer simulation of children's arithmetic word-problem solving

DENISE DELLAROSA
University of Colorado, Boulder, Colorado

ARITHPRO is a computer simulation of children's arithmetic word-problem solving behavior. It is an instantiation of a recently proposed cognitive model of the knowledge and procedures required to solve such problems. The program solves word problems by (1) comprehending the story text in which the problem is embedded, (2) comprehending numerical information as sets of objects, (3) building superstructures from these sets, thereby specifying their logical relations, and (4) using a counting procedure to derive the answer to the problem. This report describes ARITHPRO and its architecture and knowledge base. A few comparisons of ARITHPRO's performance with that of children are also provided.

ARITHPRO is a computer simulation of children's arithmetic word-problem solving behavior. It is the third stage of development in a project aimed at instantiating the Kintsch and Greeno (1985) model of children's problem-solving processes as a viable working program. ARITHPRO's first- and second-stage predecessors are WORDPRO (Fletcher, 1985) and SOLUTION (Dellarosa, 1985), respectively. The differences among the three programs lie primarily in the complexity of the memory structures they construct, the sophistication of their linguistic knowledge, and the number and types of problem-solving strategies encoded in their rule bases.

The primary aim of this project was to test the sufficiency of the Kintsch and Greeno model as a viable model of problem solving, an aim that has been adequately satisfied in that all three computerized versions of the model can in fact solve standard word problems at the approximate level of a third grade child. The second aim, and the one to which the present work is specifically addressed, was to explain regularities in the data gleaned from children's solution attempts, particulary frequently occurring errors. We hoped to produce an explicitly coded theory of children's cognitive processing during problem-solving attempts.

OVERVIEW

Kintsch and Greeno (1985) argued that children's problem-solving behaviors result from an interaction of text comprehension processes and arithmetic problem-solving strategies. More specifically, this means that the problem representation constructed by the child during a solution attempt is a joint product of his or her (1) linguistic sophistication and (2) grasp of logical set relations. The child's choice of (3) an arithmetic or other solution strategy to use in solving the problem is strictly dictated by the quality of the problem representation. In other words, a child's solution strategy is determined by how well he or she understands the problem described by the text.

These ideas were taken quite literally when designing the simulation. ARITHPRO's problem-solving style is characterized by an interaction of the three components referred to above. ARITHPRO comprehends the story described in a word problem by building proposition frames (van Dijk & Kintsch, 1983). It comprehends numerical information by building set frames; that is, it understands numbers as sets of objects. The problem structure, or more precisely, the logical relations among the sets created during comprehension, is captured in superschemata which subsume these individual sets. The particular type of superschema that is built depends upon the information present in the individual proposition and set frames. Finally, the presence of a superschema triggers an arithmetic counting procedure appropriate to the superschema, thereby producing an answer to the problem. If no superschema, or an incomplete superschema, has been created (i.e., if ARITHPRO did not "understand" the problem), then default strategies are used to produce intelligent guesses about the answer.

ARCHITECTURE: A MORE DETAILED LOOK

ARITHPRO's programming architecture includes a lexicon, written in LOOPS, and a production system, written in INTERLISP. The program runs on a XEROX 1108 "Dandelion." Embedded in this architecture are the main components of the psychological model upon which it is based, namely (1) linguistic comprehension processes, (2) knowledge concerning sets and their logical relations, (3) arithmetic counting strategies, (4) default solution

This work was supported by National Science Foundation Grant No. BNS-8309075 to Walter Kintsch and James G. Greeno. Reprint requests should be addressed to: Denise Dellarosa, Department of Psychology, University of Colorado, Boulder, CO 80309. Also available are the documentation to the program and floppy diskettes containing the program, its lexicon, and the problems it solves.

strategies, (5) a goal stack, (6) a short-term memory (STM) buffer, and (7) a long-term memory (LTM). The main data structures created by ARITHPRO are frames (Winston & Horn, 1981). A frame constitutes a single unit in STM. The buffer size of STM is five units.

How It Works

ARITHPRO currently has no parser. It takes as its input a propositionalized problem text, an example of which is illustrated in Figure 1. To begin a processing cycle, ARITHPRO loads the first set of propositions (corresponding to the first sentence) into STM and sets the current goal in "Read." A collection of functions then cycle through the rules contained in the rule base, looking for rules whose conditions are satisfied by the contents of STM and the contents of the goal stack. The first rule whose conditions are found to be satisfied is fired. Firing a rule causes memory structures to be created in STM, changed, or retrieved from LTM, and goals to be added or deleted from the goal stack. When the size of STM reaches five units, new incoming units begin to displace old ones. The displacement occurs on a modified first-in-first-out basis, with set frames being given higher priority than proposition frames. When no more rules are satisfied by the contents of STM, STM is purged by transferring to LTM all units except the most current set and proposition frames. The next set of propositions is then loaded into STM and the process begins again. The processing of a proposition containing the words HOW MANY changes the goal from "Read" to "Solve." At this point, propositions can still be processed (i.e., read), but rules governing solution procedures become available for firing (these rules have a "Solve" goal as a primary condition). Rule firing continues until no more rule conditions are satisfied, and no more sentences are left to read. Usually, ARITHPRO has derived an answer by this point.

A few characteristics of ARITHPRO should be noted. First, a cycle in this system is counted as in Kintsch and van Dijk (1978): A new cycle begins when new unprocessed propositions are loaded into STM from the problem text. Second, no conflict resolution strategies are employed or needed. The rules are ordered according to what is believed to be a psychologically valid scheme, and the first rule to be encountered whose conditions are satisfied fires. The rule ordering, combined with the goal stack, has proved to be sufficient to control processing episodes.

A Closer Look at the Rules and Lexicon

ARITHPRO's knowledge is encoded in a lexicon and a rule base, the latter of which is divided into four parts. These four parts contain rules pertaining to linguistic knowledge, knowledge about set relations, arithmetic counting procedures, and default solution strategies. A brief description of each of these follows.

Linguistic knowledge. ARITHPRO's linguistic knowledge is encoded in its lexicon and the first set of production rules. Together, they represent three types of linguistic understanding. Knowledge about words and their meanings is contained in the lexicon. Knowledge about propositions and their meanings is encoded in production rules that create proposition frames as described by van Dijk and Kintsch (1983). Knowledge about text structure is encoded in rules that create expectations (i.e., goals) concerning incoming propositions and their relations to propositions already processed.

The *lexicon* consists of words children are likely to encounter in word problems. The words belong to classes that are arranged in an inheritance hierarchy, subclasses inheriting properties from superclasses. The lexicon is depicted in Figure 2. The *proposition production rules* are triggered by the presence of words belonging to certain classes in the lexicon. When triggered, the rules create frames out of the propositions containing these words. The actions of these rules represent activation of knowledge concerning word meanings and the meanings of the propositions comprising the words. The general form of these rules is:

```
IF    This item is a proposition
      and the predicate of this proposition is a
        certain word type
```

a

```
HORTENSE HAS FIVE DIMES.
BUBBLES HAS THREE DIMES.
HOW MANY DIMES DO THEY HAVE ALTOGETHER?

(((P1 (EQUAL X HORTENSE))
  (P2 (HAVE X P3))
  (P3 (FIVE DIMES)))
 ((P4 (EQUAL Y BUBBLES))
  (P5 (HAVE Y P6
  (P6 (THREE DIMES)))
 ((P7 (HOWMANY DIMES))
  (P8 (HAVE-ALTOGETHER X&Y P7))))
```

b

```
ROLLO HAD FOUR MARBLES.
THEN NORTON GAVE HIM SIX MARBLES.
HOW MANY MARBLES DOES ROLLO HAVE NOW?

(((P1 (EQUAL X ROLLO))
(P2 (HAVE X P4))
(P3 (PAST P2))
(P4 (FOUR MARBLES)))
((P5 (THEN P2 P7))
(P6 (EQUAL X NORTON))
(P7 (GIVE Y X P9))
(P8 (PAST P7))
(P9 (SIX MARBLES)))
((P10 (HOWMANY MARBLES))
(P11 (HAVE X P10))
(P12 (NOW P11))))
```

Figure 1. An example of a COMBINE problem (a) and a TRANSFER problem (b), and their propositionalized form.

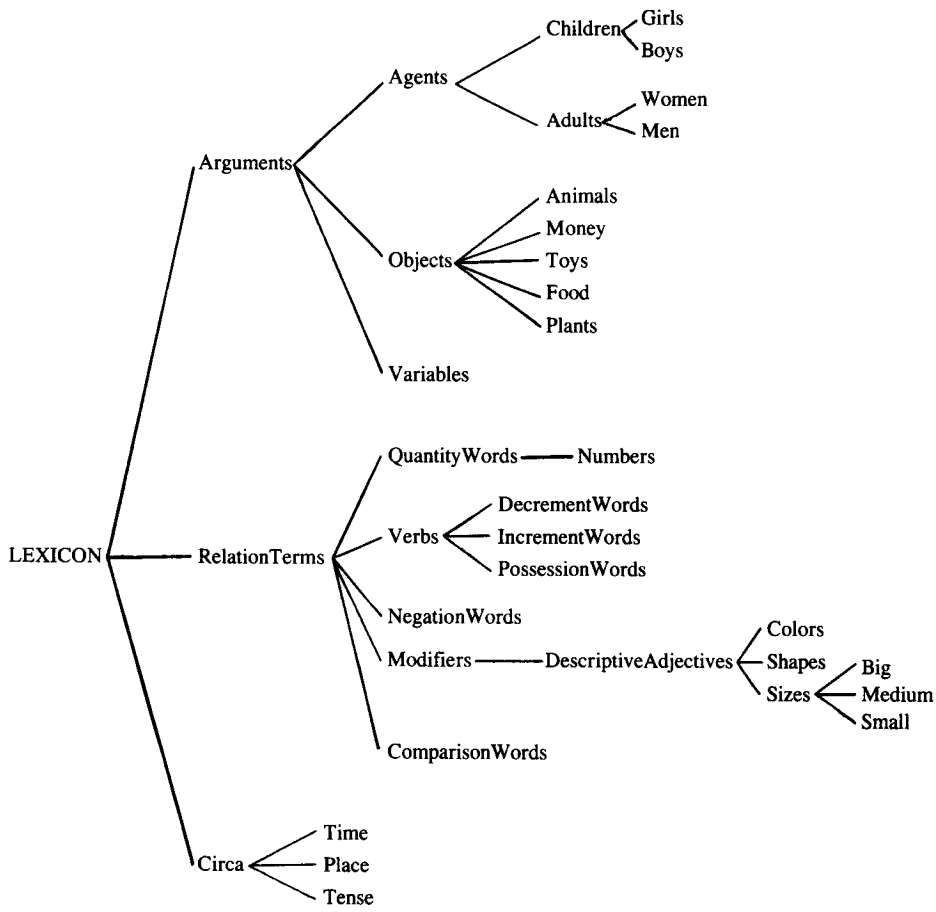

Figure 2. The hierarchical structure of ARITHPRO'S lexicon. This is an inheritance network in which word classes inherit attributes from those above them on the hierarchy.

THEN Add to STM a skeletal proposition frame
Bind the predicate to the appropriate frame slot
Bind the argument(s) to the appropriate frame slot(s)
Add class membership and any other meaning information concerning these words to the appropriate frame slots
Delete the unprocessed proposition from STM

Figure 3 illustrates an unprocessed possessive proposition taken from Figure 1a and its processed proposition frame. Note that class membership information is included in the frame. In addition, whenever a proposition references another proposition, as this one does, the two proposition frames are linked together in memory.

The remaining linguistic rules generate goals concerning set specification information and problem structure. These rules are triggered by the presence of certain types of proposition frames in STM. For example, when a possessive proposition frame is encountered, a goal is placed on the goal stack to bind ownership information to a set of objects. In other words, whenever ARITHPRO processes a proposition such as (HAVE X P3), it understands that X possesses something, that the thing that X possesses will be described shortly, and that it is likely to be a set of objects, because this is a word problem. In addition, a goal is added requesting the assignment of UNKNOWN to the role slot of the set to be encountered.

Some proposition frames carry more information concerning the problem structure than do possessive frames. For example, when a transfer proposition frame is encountered, goals are added to the stack concerning the assignment of TRANSFERSET to the set objects to be encountered, the assignment of PriorSet to the set already in existence, and the need to create a transfer superschema. At this point, ARITHPRO already has a pretty good handle on what type of problem it is dealing with—unlike the possessive frame case. This situation is illustrated in Figure 4, using P7 from Figure 1b.

Quantity propositions also create special goals. Whenever a quantity proposition, such as (THREE MARBLES) or (SOME APPLES), is processed by proposition production rules, not only is a proposition frame constructed from that proposition, but a goal to make a new set is added to the goal stack. The processing of this goal is described in the next section.

Unprocessed proposition. (P2 (HAVE X P3))

Processed proposition frame	Goal Stack
(P2 (PREDICATE : (HAVE) (ISA (PossessionWord, Verb, RelationTerm))) (ARGUMENT1 : (X) (ISA (Variable, Agent))) (ARGUMENT2 : (P3)))	((BindOwner X P3) (BindRole UNKNOWN P3) (Read))

Figure 3. An unprocessed possessive proposition and its processed proposition frame. Also shown is the goal stack produced by processing this type of proposition.

Knowledge about sets and set relations. The second group of rules are concerned with building problem structures. These *superschema production rules* divide themselves into two groups: The first deals with the comprehension of quantity words as sets of objects, and the second deals with the construction of organized problem structures, or superschemata.

Whenever ARITHPRO processes a quantity proposition, it creates a goal requesting that a new set be constructed. The presence of this goal triggers a rule that creates a skeletal set frame from the quantity proposition frame. The form of this rule is:

IF The goal is MAKESET
THEN Add to STM a skeletal set frame
Bind the quantity to the QUANTITY slot
Bind the objects to the OBJECT slot
Pop the goal

At this point, the goal stack usually contains several other goals concerning the assignment of values to the new set's SPECIFICATION slot, these goals having been generated by the proposition rules, as described above. More particularly, these specifications may include the set's OWNER, ROLE, TIME of existence (e.g., past, present), and LOCATION. Included in this set of rules are ones that deal with the assignment of specified values to facets of the SPECIFICATION slot. Continuing with our two examples, Figure 5 illustrates the complete sets created during processing of the first two lines of the TRANSFER and the COMBINE problem text from Figure 1. Notice that in the TRANSFER case, the role of TRANSFERSET has been assigned to the role slot of the set referred to in the transfer proposition frame and the role of PriorSet has been assigned to the set that was in existence prior to the transfer of goods. All of these assignments represent the actions of the rules in this group.

ARITHPRO understands four types of structures: SUPERSET, TRANSFER-IN, TRANSFER-OUT, and COMPARE. A SUPERSET structure is one in which two sets are subsets of a third set, as in our COMBINE example. A TRANSFER-IN structure is one in which more objects are added to an already existing set of objects, as in our TRANSFER example. A TRANSFER-OUT structure is one in which objects are removed from an already existing set of objects, as in

Clarissa had four teddy bears.
She gave two of them to Chatworth.
How many teddy bears does Clarissa have now?

Finally, a COMPARE structure is one in which sets are compared and the difference in their cardinality noted, as in

Hortense has twelve records
Theodore has ten records.
How many more records does Hortense have than Theodore?

Each of these four structures is encoded in ARITHPRO's knowledge base as a superschema whose slots must be filled with sets having certain specifications. Their creation in STM is triggered by certain proposition frames, or by the combination of certain proposition frames and set frames having certain specifications. In our TRANSFER example, the following rule would apply:

IF The goal is MAKE-TRANSFER-SCHEMA
and the owner of the PriorSet is the
patient of the transfer proposition
THEN Make a skeletal TRANSFER-IN superschema frame
in STM
Add a goal to find the TRANSFERSET
Add a goal to find the STARTSET
Add a goal to find the RESULTSET

The existence of a transfer proposition frame, such as that illustrated in Figure 4, triggers the addition of a goal calling for the creation of a TRANSFER superschema to the goal stack. This goal, combined with a prior set whose owner was the patient of the transaction, triggers the creation of a TRANSFER-IN superschema. (If PriorSet's owner had been the agent, a TRANSFER-OUT superschema would have been constructed.) New goals calling for the three sets that complete a TRANSFER-IN schema are added to the goal stack. Two of these—TRANSFERSET and STARTSET—can be filled immediately because sets answering the specifications of TRANSFERSET and STARTSET currently reside in STM. The third goal—RESULTSET—is filled once the third line of the problem is processed, thereby producing a set with the required specifications. Specification matching and slot

Unprocessed proposition: (P7 (GIVE Y X P9))

Processed proposition frame	Goal Stack
(P7 (PREDICATE : (GIVE) (ISA (TransferWord, Verb, RelationTerm))) (ARGUMENT1 : (Y) (ISA (Variable, Agent)) (ROLE : (Agent)) (ARGUMENT2 : (X) (ISA (Variable, Agent)) (ROLE : (Patient)) (ARGUMENT3 · (P9)))	((BindOwner X P3) (BindRole TRANSFERSET P3) (BindRole PriorSet ExistingSet) (MakeTransferSchema) (Read))

Figure 4. An unprocessed transfer proposition and its processed proposition frame. Also shown is the goal stack produced by processing this type of proposition.

a

```
(SET1  (QUANTITY : 5)
       (OBJECTS : DIME)
       (ROLE : UNKNOWN)
       (SPECIFICATION
          (OWNER : X)))

(SET2  (QUANTITY : 3)
       (OBJECTS : DIME)
       (ROLE : UNKNOWN)
       (SPECIFICATION
          (OWNER : Y)))
```

b

```
[TransferInSET
  (STARTSET
    (SET 1 (QUANTITY : 4)
           (OBJECTS : MARBLE)
           (ROLE :PriorSet)
           (SPECIFICATION
             (OWNER : X)
             (TIME : PAST))))

  (TRANSFERSET
    (SET2 (QUANTITY : 6)
    (OBJECTS : MARBLE)
    (ROLE : TransferSet)
    (SPECIFICATION
      (OWNER : X)
      (TIME : PAST, AFTER SET1)))))]
```

Figure 5. The set structures constructed after processing the first two lines of a COMBINE-type problem (a) and a TRANSFER-type problem (b).

filling are, of course, also handled by superschema production rules (for more information, see Dellarosa, 1985).

Counting procedure and default rules. The remaining two groups of rules in ARITHPRO's knowledge base are *procedure rules*, which deal with counting strategies for deriving answers, and *default rules*, which constitute best guesses at problem solutions when the frames ARITHPRO has constructed fit none of its superschemata. The procedure rules are not described here; the interested reader is directed to Dellarosa (1985) for more information. The default rules fall into two groups. The first deals with searching STM or LTM for sets with certain specifications and returning their cardinality as best guesses. The second deals with mining proposition frames in STM and LTM for helpful verbal information with which to disambiguate the set structures. These rules are discussed more fully in the next section, where illustrative examples make their purposes and functions clearer.

ARITHPRO's PERFORMANCE AND THAT OF CHILDREN

Two questions are addressed in turn: What types of problems can ARITHPRO solve and what are its limitations? How do ARITHPRO's problem-solving characteristics compare with those of children?

Capabilities and Limitations

ARITHPRO can reliably solve the 14 problem types described by Riley, Greeno, and Heller (1983). These fall into the four categories captured by ARITHPRO's superschemata. These are the same problems that WORDPRO was designed to solve; however, unlike WORDPRO, ARITHPRO does not operate on a key-word basis, and hence can tolerate a great deal of variability in the wording of these problems. For example, WORDPRO looks exclusively for key words such as GIVE or HAVE-MORE-THAN in solving problems. Because ARITHPRO deals with word classes, any transfer word or comparison word will produce the same problem-solving sequences.

In addition to these 14 problem types, ARITHPRO can solve problems that its predecessors could not. Table 1 contains three examples of such problems. In the first, ARITHPRO creates a SUPERSET structure, operating on the premise that the least specified of three sets whose specification slots overlap is the superset. Thus, the third set must be the superset: all three sets have matching objects (WINDOWS) and are located in the same place (HALL), but the third set does not have a size specification and the other two do. Importantly, the disparate information in a specification slot must indicate disjointness between the two subset candidates. For example, if the second line of the problem stated "and four ROUND windows in the hall," ARITHPRO would not attempt to solve this as a SUPERSET problem because the first two sets, although both subsets of the third, are not disjoint. There could be big, round windows.

The remaining two problems also describe SUPERSET problem structures. In Problem 2, a conjunction rule is used to determine the superset. This rule states that if one set specifies a conjunction of some aspect of the other two sets (in this case OBJECTS), then the conjunction set must be the superset and the other two subsets. Problem 3 describes a case in which a lexicon is used to determine the superset. In this case, DOLLS and TEDDY BEARS are both members of the class TOYS. ARITHPRO uses superclass-subclass and class-instance information from its lexicon to identify which of the three sets is the superset.

Table 1
Three Sparsely Worded Problem Types that ARITHPRO Can Solve

1	There are three large windows and four small windows in the hall. How many windows are there in the hall?
2.	There are four dolls and three teddy bears on the shelf. How many dolls and teddy bears are there on the shelf?
3.	There are four dolls and three teddy bears on the shelf. How many toys are there on the shelf?

Comparison with Children's Performance

ARITHPRO, like its predecessors, is designed to simulate optimal performance on word problems. Its rules capture empirically observed performance characteristics of children who successfully solved the various problems. However, children are not always successful at solving problems—they make mistakes. Moreover, there is a great deal of regularity among the errors children commit, and these characteristic errors tend to decrease as children grow older (Riley et al., 1983). In comparing ARITHPRO's performance with that of children, we have chosen to focus on the errors children commit and the modifications required to simulate these errors.

This decision was motivated by the following fact: A tension exists in the literature between theorists who believe that developmental changes in performance are due to knowledge acquisition or maturation in reasoning powers and those who believe that such changes in performance are due to improved linguistic skills on the part of the child. The former camp includes Piagetian researchers whose view is based on empirical observations of developmental differences in performance on carefully chosen tasks. Their argument is that if children under a certain age typically fail at a task that requires certain knowledge to perform—such as a conversation task—then the children who fail do not yet possess the requisite knowledge for that task. Researchers in the second camp base their view on studies in which minor wording changes in task instructions have produced dramatic performance improvements in children who typically failed the task under standard conditions (e.g., Hudson, 1983; Markman, 1979). Their argument is that children may possess the requisite knowledge to perform a task, but may misinterpret or misunderstand the complex linguistic utterances used by adults to describe the task, and hence fail to perform it correctly.

Simulations such as ARITHPRO provide a useful means of exploring this controversy. Unlike children, ARITHPRO's knowledge can be deleted, changed, or added at will, and the effects of these changes on performance noted. In particular, the different effects of deleting or changing linguistic and mathematical knowledge can be easily compared.

In taking this approach, our initial question has been: How many characteristic errors can ARITHPRO be induced to commit simply by manipulating its linguistic knowledge and/or input? We are particularly interested in how far this linguistic approach can be taken before it becomes necessary to begin removing mathematical/set knowledge in order to produce the errors that children make. We have just begun this approach, and the results are thus far encouraging.

A particularly hard problem type for children to solve is one involving the word *some*, as in

Rufus had some marbles.
Then Gertrude gave him eight more marbles.
Now Rufus has thirteen marbles.
How many marbles did Rufus have in the beginning?

The most frequently committed error on this type of problem is to give as the answer the cardinal of the transfer set, in this case eight. It has been suggested that this error is caused by a failure on the part of the child to understand that things can be undone or done backward in time (Briars & Larkin, 1984). This view, therefore, suggests that young children lack a type of knowledge concerning manipulation of objects in time. However, we suggest that the clue lies in the way children interpret the word *some*. Riley et al. (1983) noted that, when using blocks to model story problems, children often were at a loss as to how to model *some*. Often they simply ignored the entire sentence containing this word in their solution attempts. To simulate this situation, we ran ARITHPRO twice on this problem, with *some* entered once as a member of the quantity word class and once as a modifier. The output from both of these trials is depicted in Figure 6. When ARITHPRO understood *some* as a quantity word, it built the correct TRANSFER-IN structure with the STARTSET cardinal as the goal, as shown in Figure 6a. When ARITHPRO understood *some* as a modifier, however, it did not build a TRANSFER-IN structure, but instead ended up with three individual sets, as depicted in Figure 6b. To understand why this happened, recall that the construction of a TRANSFER-IN structure requires the existence of a transfer proposition frame and a prior set whose owner is the patient of the transfer proposition frame. Because *some* was not recognized as a quantity word, ARITHPRO did not build a set to represent it. The second line of the problem specified a TRANSFERSET, but because no prior set existed, the TRANSFER-IN rule did not fire. Finally, the third set, specified by HOWMANY, was constructed, but still no superstructure information was encountered in the problem text. Because none of its standard rules applied to this memory configuration, ARITHPRO resorted to its default rules in an attempt to solve the problem. Several of these rules involve mining the text-base proposition frames for a clue or key word that will help solve the problem. In this case, ARITHPRO found the proposition frame containing BEGINNING and interpreted this as "RETURN THE CARDINAL OF THE FIRST SET CREATED." This is, of course, SET1, the TRANSFERSET.

Another notoriously difficult problem for children to solve involves a comparison, as in

Larry has seven oranges.
He has three more oranges than Jeff.
How many oranges does Jeff have?

The most typical error committed on this type of problem is to give as the answer the cardinal of the difference set, in this case three. The source of this error is not known. What is known, however, is that children have enormous difficulty with the comparative linguistic form (e.g., *have more than, have less than*), and interpret it in a variety of ways. To simulate this situation, we ran ARITHPRO with two different parsings of the second sentence. In the

a

```
[TransferInSET
  (STARTSET
    (SET1 (QUANTITY : GOAL)
          (OBJECTS : MARBLES)
          (ROLE :PriorSet)
          (SPECIFICATON
            (OWNER : X)
            (TIME : PAST))))

(TRANSFERSET
    (SET2 (QUANTITY : 8)
          (OBJECTS : MARBLES)
          (ROLE : TransferSet)
          (SPECIFICATION
            (OWNER : X)
            (TIME : (PAST,
                  AFTER SET1))))

(RESULTSET
    (SET3 (QUANTITY : 13)
          (OBJECTS : MARBLES)
          (ROLE : ResultSet)
          (SPECIFICATION
            (OWNER : X)
            (TIME : PRESENT))))]
```

b

```
(SET1  (QUANTITY : 8)
       (OBJECTS : MARBLES)
       (ROLE ; TransferSET)
       (SPECIFICATON
         (OWNER : X)
         (TIME : (PAST, AFTER NIL))))

SET2   (QUANTITY : 13)
       (OBJECTS : MARBLES)
       (ROLE : ResultSet)
       (SPECIFICATION
         (OWNER : X)
         (TIME : PRESENT)))

(SET3  (QUANTITY : UNKNOWN)
       (OBJECTS : MARBLES)
       (ROLE : UNKNOWN)
       (SPECIFICATION
         (OWNER : X)
         (TIME : (PAST, BEGINNING))))
```

Figure 6. The set structures constructed from a TRANSFER problem when SOME is classified as a quantity word (a) and as a modifier (b).

first (correct) case, the second line was parsed as

(P4 (EQUAL Y JEFF))
(P5 (HAVE-MORE-THAN X Y P6))
(P6 (THREE ORANGES))

In the second (incorrect) case, the second line was parsed as

(P4 (EQUAL Y JEFF))
(P5 (HAVE Y P7))
(P6 (MORE-THAN P3 P7))
(P7 (THREE ORANGES))

In the second case, the problem states that JEFF has three oranges, and the seven oranges is MORE-THAN three oranges. The MORE-THAN is just a bit more information about the relation between the cardinal; it is no longer a specification of the difference set in the problem. Note also that the ownership of this set is incorrect. The problem states that LARRY owns the second set, but the owner here is specified as JEFF.

The output from the two cases is illustrated in Figure 7. In the first case, ARITHPRO encountered the comparison proposition and built a comparison proposition frame. This triggered a goal to build a COMPARE superschema. The three sets were then assigned to the appropriate slots, and the problem was solved using a matching procedure. This is depicted in Figure 7a. In the second case (Figure 7b), however, ARITHPRO built the three sets specified in the problem but did not build a COMPARE superschema, because the MORE-THAN proposition was treated as a description of a comparison between two cardinalities, and not as a carrier of information about the degree of difference between the two sets. Once again, because no superschemata were built, ARITHPRO had to resort to its default rules to try to solve the problem. In this case, no key word or other type of information was present to disambiguate the situation. Another rule did apply, however, one that instigated a search of the set structures to determine whether the requested information was already known. Note that the last line of the problem requests the cardinal of the set owned by JEFF. This information was present in STM, and it was returned as the answer. This was the cardinal of the difference set.

In both of these situations, ARITHPRO possessed the set/mathematical knowledge required to solve the problem, but that knowledge was not activated due to faulty linguistic information. In these cases, therefore, it was not necessary to remove set/mathematical knowledge to produce characteristic errors. Investigation of other error types is currently under way.

Finally, we have also begun to use ARITHPRO's output to make predictions concerning memory for verbal aspects of the text. The first prediction concerns the recallability of existential propositions that contain the names of the actors in the story. Unlike most of the other propositions, existential propositions that contain name information are not referenced by—and hence not linked to—any other proposition in the text. Moreover, because these are typically the first propositions processed, they are also more likely to be displaced by new incoming propositions and consequently lost from STM before the end of a cycle. Both of these characteristics suggest that names should be particularly difficult for children to recall from story. This is precisely the case. Children have been found to rapidly forget the names of the characters in story

a

```
[CompareSET
  (LARGESET
    (SET1 (QUANTITY : 7)
          (OBJECTS : ORANGES)
          (ROLE : UNKNOWN)
          (SPECIFICATION
            (OWNER : X)

  (DIFFERENCESET
    (SET2 (QUANTITY : 3)
          (OBJECTS : ORANGES)
          (ROLE : DifferenceSet)
          (SPECIFICATION
            (OWNER . X))))

  (SMALLSET
    (SET3 (QUANTITY : GOAL)
          (OBJECTS : ORANGES)
          (ROLE : UNKNOWN)
          (SPECIFICATION
            (OWNER : X)))))]
```

b

```
(SET1  (QUANTITY : 7)
       (OBJECTS  ORANGES)
       (ROLE · UNKNOWN)
       (SPECIFICATION
         (OWNER : X)

(SET2  (QUANTITY : 3)
       (OBJECTS · ORANGES)
       (ROLE : UNKNOWN)
       (SPECIFICATION
         (OWNER . X)))

(SET3  (QUANTITY : GOAL)
       (OBJECTS : ORANGES)
       (ROLE : UNKNOWN)
       (SPECIFICATION
         (OWNER · X)))
```

Figure 7. The set structures from two parsings of a difficult COMPARE problem. See text for details.

problems, even when their memory for the rest of the problem text is quite good (Dellarosa, Weimer, & Kintsch, 1985).

The second prediction concerns recallability of text-base propositions in general. Because superschemata and set structures are given higher priority than text-base frames during STM displacement, recall following a solution attempt should be lower than it would otherwise be, and recall of difficult, resource-demanding problems following a solution attempt should be strongly influenced by the schemata built, because most of the text base will have been lost or written to LTM during the attempt and may be difficult or impossible to retrieve. Both of these predictions were confirmed by Dellarosa et al. (1985). Problem recall tended to be lower following a solution attempt than under simple recall conditions, and this was particularly true of difficult problems. Moreover, recall of difficult problems seemed to be more a matter of reconstruction than of recall. For example, the superstructures built from an easy compare and a difficult compare problem are identical, but the text base differs dramatically. The comparable Compare-EASY problem for the above Compare-HARD problem would have as its second line

Jeff has three fewer oranges than Larry.

Note that this has the same problem structure as Compare-HARD, but with different wording. Dellarosa et al. (1985) found that children tended to find Compare-HARD significantly more difficult to solve than Compare-EASY, but if they had solved it correctly they were just as likely to recall the problem as Compare-EASY as Compare-HARD. In other words, they were equally likely to recall the second line as "Jeff has three fewer oranges than Larry" or as "He has three more oranges than Jeff," even though they had heard the latter and not the former. The authors interpreted this indifference to the text base as an indication that children were reconstructing the story from the problem structures they had built, and were simply guessing at the verbal structure of the second line, knowing that both forms were used in the stimulus materials. Moreover, this indifference was noted almost exclusively following a solution attempt; in the recall-only condition, the recalled verbal structure was far more likely to mirror the structure actually heard.

REFERENCES

BRIARS, D. J., & LARKIN, J H. (1984). An integrated model of skill in solving elementary word problems. *Cognition & Instruction*, **1**, 245-296.

DELLAROSA, D. (1985). *Solution: A computer simulation of children's arithmetic word-problem solving* (Tech Rep. No. 148). Boulder. University of Colorado, Institute of Cognitive Science.

DELLAROSA, D., WEIMER, R., & KINTSCH, W. (1985). *Children's recall of arithmetic word problems* (Tech. Rep No. 148). Boulder: University of Colorado, Institute of Cognitive Science.

FLETCHER, C. R. (1985). Understanding and solving arithmetic word problems: A computer simulation. *Behavior Research Methods, Instruments, & Computers*, **17**, 565-571

HUDSON, T. (1983). Correspondences and numerical differences between disjoint sets. *Child Development*, **54**, 85-90.

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

KINTSCH, W., & VAN DIJK, T. (1978). Toward a model of text comprehension and production *Psychological Review*, **85**, 363-394.

MARKMAN, E. M. (1979) Classes and collections: Conceptual organization and numerical abilities. *Cognitive Psychology*, **11**, 395-411.

RILEY, M. S., GREENO, J G., & HELLER, J. I. (1983). Development of children's problem-solving ability in arithmetic. In H. Ginsberg (Ed.), *The development of mathematical thinking*. New York: Academic Press.

VAN DIJK, T., & KINTSCH, W. (1983). *Strategies of discourse comprehension*. New York: Academic Press.

WINSTON, P. H., & HORN, B. K. P. (1981). *LISP*. Reading, MA: Addison-Wesley.