Behavior Research Methods, Instruments, & Computers
1985, Vol. 17 (5), 565-571

COMPUTER SIMULATION

Understanding and solving arithmetic word problems: A computer simulation

CHARLES R. FLETCHER
University of Minnesota, Minneapolis, Minnesota

In this paper, I describe a computer program, WORDPRO, which simulates the psychological processes involved when third-grade children understand and solve simple arithmetic word problems. Both the implementation of the program and its performance on a set of sample problems are presented. WORDPRO is a useful research tool in that it demonstrates the sufficiency of the theory upon which it is based, assists in communicating that theory to other researchers, and provides a sources of empirical predictions for experimental tests of the theory.

WORDPRO, a computer program written in Interlisp-D, implements Kintsch and Greeno's (1985) theory of the comprehension and solution of simple arithmetic word problems. The program is intended to demonstrate the sufficiency of that theory, to assist in communicating it to other researchers, and to serve as a tool for exploring the theory's consequences. In this paper, I address each of these goals. I describe the behavior of WORDPRO on a set of sample problems, show how empirical predictions are derived from it, and provide enough details of its implementation to allow other researchers to understand its output and (with the internal documentation) to test and modify it to meet their needs.

THE DOMAIN OF WORDPRO

WORDPRO is designed to simulate the comprehension and solution of a restricted set of word problems by third-grade children. Examples of the problems are shown in Table 1. Each problem can be solved by a single addition or subtraction operation.

WORDPRO currently does not accept as input the surface representation of a text. Instead it begins with a set of propositions that represent the text's meaning (see, e.g., Bovair & Kieras, 1985; Turner & Greene, 1978). As an example, Table 2 shows the propositional representation of the first text from Table 1.

KNOWLEDGE STRUCTURES

Given the propositional representation of a text, WORDPRO constructs a bilevel representation, which it then uses to derive the solution. The two levels are referred to as the text base and the problem model. The text base is an organized set of propositions. Such a representation is assumed to result from reading or listening to any text, regardless of the domain. The problem model is a nonpropositional, domain-specific representation which drives the problem-solving process (van Dijk & Kintsch, 1983, chap. 10).

Knowledge of sets is assumed to underly construction of both the text base and the problem model. This knowledge is captured by a set schema with three slots: an object slot that holds the intension of a set (e.g., marbles), a quantity slot that contains its cardinality (3), and a specification slot that holds information that distinguishes one set from another. At present, WORDPRO's set schemata include specification subslots for owner (JOE) and time (past). Other subslots, such as location, can be added easily. In the test base, the slots of the set schema are filled with propositions. In the problem model, all information not essential to a slot is stripped away. As an example, if the object slot in the text base were filled with the proposition (P4 (3 MARBLE)), the same slot in the problem model would contain simply MARBLE.

Three higher level schemata are used to organize set schemata into coherent problem representations. A transfer schema captures the relationship between a start set, a transfer set, and a result set. A superset schema is used to organize a superset and two subsets. Finally, a more-than (or less-than) schema describes the relationship between a small set, a large set, and a difference set.

WORDPRO's schemata and the procedures that store and retrieve information from them are adapted from Winston and Horn (1981, chap. 22).

FLOW OF CONTROL

The basic structure of WORDPRO is shown in Figure 1. The heart of the system is an ordered set of production rules (Newell, 1973). Each rule consists of

This work was supported by National Science Foundation Grant BNS-8309075 to W. Kintsch and J. G. Greeno. I am grateful to Walter Kintsch, Denise Dellarosa, Suzanne Mannes, Bill Fox, Alan Lesgold, Joseph Sidowski, and an anonymous reviewer for their comments on an earlier draft of this paper. Reprint requests should be addressed to C. R. Fletcher, Department of Psychology, University of Minnesota, 75 East River Road, Minneapolis, MN 55455.

Table 1
Examples of the Problem Types Handled by WORDPRO

CHANGE:	Result Unknown
	1. Joe had three marbles. Then Tom gave him five marbles. How many marbles does Joe have now?
	2. Fred had five boxes. Then he gave two boxes to Susan. How many boxes does Fred have now?
	Change Unknown
	3. Mary had four pencils. Then Sam gave her some pencils. Now mary has nine pencils. How many pencils did Sam give Mary?
	4. Betty had eleven puppies. Then she gave Debby some puppies. Now Betty has four puppies. How many puppies did Betty give Debby?
	Start Unknown
	5. Ed had some fish. Then Alan gave him eight fish. Now Ed has thirteen fish. How many fish did Ed have in the beginning?
	6. Bob had some hats. Then he gave four hats to Danny. Now Bob has eight hats. How many hats did Bob have in the beginning.
COMBINE:	Superset Unknown
	1. Lucy has two dimes. Sarah has six dimes. How many dimes do they have altogether?
	Subset Unknown
	2. Larry and Sally have twelve kittens altogether. Larry has three kittens. How many kittens does Sally have?
COMPARE:	Difference Unknown
	1. Dan has six books. Jill has two books. How many books does Dan have more than Jill?
	2. Dennis has eleven apples. Fred has six apples. How many apples does Fred have less than Dennis?
	Compared Quantity Unknown
	3. Liz has four dollars. Sarah has three dollars more than Liz. How many dollars does Sarah have?
	4. Anne has nine flowers. Kathy has five flowers less than Anne. How many flowers does Kathy have?
	Referent Unknown
	5. Larry has seven oranges. He has three oranges more than Jeff. How many oranges does Jeff have?
	6. Alice has six horses. She has five horses less than Doris. How many horses does Doris have?

Table 2
Propositional Representation of the Change-1 Text From Table 1

(Change-1	(((P1	(EQUAL X JOE))
	(P2	(PAST P3))
	(P3	(HAVE X P4))
	(P4	(3 MARBLE)))
	((P5	(THEN P3 P7))
	(P6	(EQUAL Y TOM))
	(P7	GIVE Y X P9))
	(P8	(PAST P7))
	(P9	(5 MARBLE)))
	((P10	(HOWMANY MARBLE))
	(P11	(HAVE X P10))
	(P12	(NOW P11)))))

a set of conditions which must be met for the rule to be applied, and a set of actions which are then carried out. Each condition is a test on the current contents of short-term memory (STM). The actions, in turn, alter the state of STM by adding new information, reinstating old information from long-term memory (LTM), or deleting information. Thus, we assume that the set of production rules controls the flow of information through STM. The rules are checked in order against the contents of STM. When more than one rule matches the contents of STM, only the first one encountered is applied. To make WORDPRO a useful tool for communicating the theory upon which it is based, it is designed so that the user can watch STM change rule by rule. Before each rule is applied, the user is shown its name and the current contents of STM.

As Figure 1 also shows, the production rules are applied within a simple read-purge loop, giving WORDPRO the cyclical character necessary in a model of text comprehension (Fletcher, 1981; Kintsch & van Dijk, 1978). During the read portion of the loop, those propositions corresponding to the next sentence are added to STM. The production rules then operate on the contents of STM until no rules apply. At this point, STM is purged of all but one to three items, and the cycle begins again. This prevents STM from becoming overloaded but necessitates occasional searches of LTM for purged information. Two types of information are not purged: the most recently constructed set schema or higher level schema and requests by higher level schema for set schemata to take up unfilled slots.

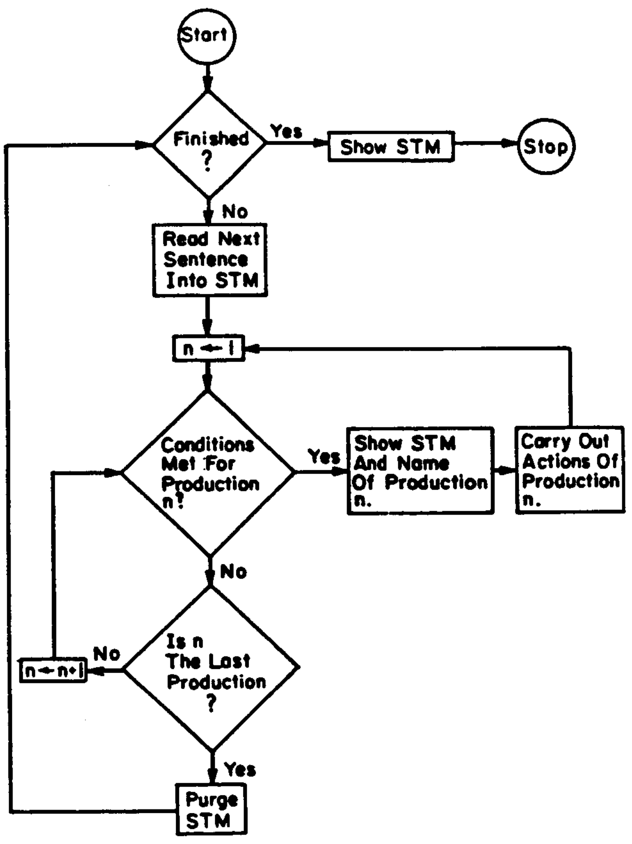
Start
Finished ?
Yes
Show STM
Stop
No
Read Next Sentence Into STM
n ← 1
Conditions Met For Production n?
Yes
Show STM And Name Of Production n.
Carry Out Actions Of Production n.
No
Is n The Last Production ?
No
n ← n+1
Yes
Purge STM

Figure 1. The basic control structure of WORDPRO.

WORDPRO'S PRODUCTION RULES

WORDPRO's production rules fall into three categories: meaning postulates, arithmetic strategies, and problem-solving procedures. The first 13 rules are the meaning postulates. Each of these is triggered by the occurrence of a proposition with a particular predicate (first term) which has not been added to the text base. A postulate's primary action is to add that proposition to the text base and corresponding information to the problem model. Two additional actions are carried out by subsets of these rules. Quantifying propositions (i.e., those with SOME, HOWMANY, or a number as their predicate) cause a new set schema to be added to STM. Propositions with GIVE, HAVE-ALTOGETHER, HAVE-MORE-THAN, or HAVE-LESS-THAN as their predicate initiate searches of LTM under some conditions. English approximations of some representative meaning postulates are given in Table 3. As the table indicates, each rule is given a mnemonic name. These are the names shown to the program's user before each rule is applied.

The arithmetic strategies construct higher level schemata. Their actions include adding new schemata to STM, filling the slots of those schemata, and adding requests to STM. Each request is for a set schema with a given specification which is needed to complete a higher level schema. The conditions which trigger these rules are either set schemata with certain specifications, or the simultaneous occurrence of requests and set schemata which satisfy them. WORDPRO includes 12 arithmetic strategies; 2 representative examples are shown in Table 4.

The final 11 production rules comprise the problem-solving procedures. Each of these rules is triggered by a completed higher level schema. There are two types of problem-solving procedures. Some derive solutions by performing a sequence of operations on mental blocks. The allowable operations including adding or deleting blocks from STM, counting blocks, matching blocks, and moving them from one mental location to another. The remaining rules do not actually find solutions; rather they convert one type of higher level schema into another (e.g., a transfer schema might be changed into a superset schema). Table 5 lists some representative problem-solving procedures.

The procedures that test each production rule's conditions and apply its action (when appropriate) are modifications of those described in Winston and Horn (1981, chap. 18).

Table 3
English Approximations of Two Representative Meaning Postulates

	Example 1
NAME:	HowManyProp
CONDITIONS:	1. Does STM contain a proposition?
	2. Does that proposition have the predicate HOWMANY?
ACTIONS:	1. Add a new set schema to STM.
	2. Put the proposition in the object slot of the text base.
	3. Put the second term of the proposition in the object slot of the problem model.
	4. Put GOAL in the quantity slot of the problem model.
	Example 2
NAME:	NowProp
CONDITIONS:	1. Does STM contain a proposition?
	2. Does that proposition have the predicate NOW?
ACTIONS:	1. Put the proposition in the specification slot of the text base.
	2. Put (TIME:PRESENT) in the specification slot of the problem model.

Table 4
English Approximations of Two Representative Arithmetic Strategies

	Example 1
NAME:	TransferIn
CONDITIONS:	1. Does STM contain a set schema?
	2. Does the text base of the set schema contain a proposition with GIVE in its specification?
	3. Does STM contain a second set schema with the same owner as the first one?
ACTIONS:	1. Add a new transfer schema to STM.
	2. Put the first set schema in the transfer-in slot.
	3. Put the second set schema in the start slot.
	4. Add to STM a request for a result set with the same owner as the first two sets.
	Example 2
NAME:	ResultSet
CONDITIONS:	1. Does STM contain a request?
	2. Is the request being made by a transfer schema?
	3 Is a result-set being requested?
	4. Does STM contain a set schema whose specification matches the request?
ACTIONS:	1. Put the set schema in the result slot of the transfer schema.
	2. Remove the request from STM.

Table 5
English Approximations of Two Representative Problem Solving Procedures

	Example 1
NAME:	ConvertChange5
CONDITIONS:	1. Does STM contain a transfer schema?
	2. Does the start-set of the transfer schema have GOAL as its quantity?
	3. Does the transfer-in-set have a number as its quantity?
	4. Does the result-set have a number as its quantity?
ACTIONS:	1. Change the transfer schema to a superset schema.
	2. Put the start-set in one subset slot.
	3. Put the transfer-in set in the other subset slot.
	4. Put the result-set in the superset slot.
	Example 2
NAME:	SolveChange1
CONDITIONS:	1. Does STM contain a transfer schema?
	2. Does the start-set of the transfer schema have a number (X) as its quantity?
	3. Does the transfer-in-set have a number (Y) as its quantity?
	4. Does the result-set have GOAL as its quantity?
ACTIONS:	1. Add X blocks to STM.
	2. Add Y blocks to STM.
	3. Count the blocks in STM.

AN EXAMPLE

As an example of how the program works, consider Table 6 which shows the contents of STM as WORDPRO begins to process the final sentence of the change-1 problem shown in Tables 1 and 2. At this point STM contains the propositions from the final sentence plus two chunks from earlier in the text, a transfer schema and a

Table 6
The Contents of STM at the Beginning of the Third Processing Cycle for the Change-1 Problem

```
PROPOSITIONS:

1. (P10 (HOWMANY MARBLE))

2. (P11 (HAVE X P10))

3. (P12 (NOW P11))

SCHEMATA:

4. (TRANSFER-SET

      (TRANSFER-IN

         (SET2 (MODEL (QUANTITY (5))

                      (OBJECT (MARBLE))

                      (SPECIFICATION ((OWNER:JOE)

                                      (TIME:PAST)

                                      (TIME:AFTER P3))))

               (T-BASE (OBJECT ((P9 (5 MARBLE))))

                       (SPECIFICATION ((P7 (GIVE Y X P9))

                                       (P6 (EQUAL Y TOM))

                                       (P8 (PAST P7))

                                       (P5 (THEN P3

                                                 P7)))))))

      (START

         (SET1 (MODEL (QUANTITY (3))

                      (OBJECT (MARBLE))

                      (SPECIFICATION ((OWNER:JOE)

                                      (TIME:PAST))))

               (T-BASE (OBJECT ((P4 (3 MARBLE))))

                       (SPECIFICATION ((P3 (HAVE X P4))

                                       (P1 (EQUAL X JOE))

                                       (P2 (PAST P3)))))))))

REQUESTS:

5. (REQUEST TRANSFER-SET RESULT (OWNER:JOE))
```

Note—At this point, STM contains five chunks of information.

Table 7
The Contents of STM Before the Final Production Rule is Applied to Find the Solution of the Change-1 Problem

```
SCHEMATA:

1. (TRANSFER-SET

      (TRANSFER-IN

         (SET2 (MODEL (QUANTITY (5))

                      (OBJECT (MARBLE))

                      (SPECIFICATION ((OWNER:JOE)

                                      (TIME:PAST)

                                      (TIME:AFTER P3))))

               (T-BASE (OBJECT ((P9 (5 MARBLE))))

                       (SPECIFICATION ((P7 (GIVE Y X P9))

                                       (P6 (EQUAL Y TOM))

                                       (P8 (PAST P7))

                                       (P5 (THEN P3

                                                 P7)))))))

      (START

         (SET1 (MODEL (QUANTITY (3))

                      (OBJECT (MARBLE))

                      (SPECIFICATION ((OWNER:JOE)

                                      (TIME:PAST))))

               (T-BASE (OBJECT ((P4 (3 MARBLE))))

                       (SPECIFICATION ((P3 (HAVE X P4))

                                       (P1 (EQUAL X JOE))

                                       (P2 (PAST P3)))))))

      (RESULT

         (SET3 (MODEL (QUANTITY (GOAL))

                      (OBJECT (MARBLE))

                      (SPECIFICATION ((OWNER:JOE)

                                      (TIME:PRESENT))))

               (T-BASE (OBJECT ((P10 (HOWMANY MARBLE))))

                       (SPECIFICATION ((P11 (HAVE X P10))

                                       (P12 (NOW P11))))))))
```

Note – At this point, STM contains a single chunk, the transfer schema.

request for a result set. From here processing proceeds as follows. First, three meaning postulates are triggered which build a new set schema, SET3. The first of these is triggered by (P10 (HOWMANY MARBLE)). This rule adds the new schema to STM and fills in its object and quantity slots. Next, (P11 (HAVE X P10)) activates a rule which adds owner information to the specification slot. Finally, (P12 (NOW P11)) completes the schema by filling the time subslot of the specification. This completed set schema matches the requirements of the request still active in STM. Thus, an arithmetic strategy is triggered which adds SET3 to the result slot of the transfer schema. This completes the transfer schema, as shown in Table 7, and evokes one of the problem-solving strategies. This rule arrives at the correct solution to the problem by adding two sets of mental blocks to STM, one corresponding to the start set and the other to the transfer-in set, and then by counting the total number of blocks.

Tables 6 and 7 also illustrate the format used by the program for displaying the contents of STM. In the tables, as in the program itself, each schema is represented as a list. The first list element identifies the schema. The remaining elements are themselves lists which identify each slot and the information that fills it. When that information is itself a schema, it is represented in the same way. Each request is represented as a list with four elements. The first of these identifies it as a request, the second identifies the higher level schema for which the request is being made, the third identifies the slot that needs to be filled, and the last contains information which a set schema's specification must contain for it to fill the request. Propositions are represented in Tables 2 and 6.

EVALUATION AND FUTURE DIRECTIONS

WORDPRO can be considered a success for a number of reasons. First, it successfully solves all 14 problem types for which it was designed. Second, it makes the theory underlying it easier to understand by allowing one to watch it work. Finally, it allows a test of the theory by comparing the program's performance to that of human subjects. Run-time statistics such as those shown in Table 8 form the basis for this comparison. Each of the operations in the table is assumed to be resource consuming and should increase the time required to read and solve a problem while reducing the probability of a correct answer. The same statistics are displayed during the operation of the program and are updated each time a production rule is applied.

Plans are now under way to expand WORDPRO in a number of directions. The first is to enlarge the domain of problems that the program can handle. The second is to adapt it to different levels of ability. The last is to interface WORDPRO with a parser that will allow it to accept the surface representation of a text as input.

AVAILABILITY AND USE

Researchers interested in exploring WORDPRO are encouraged to acquire a copy of the program by sending a blank 8-in. diskette to Walter Kintsch, Department of Psychology, University of Colorado, Boulder, CO 80309. The diskette will be returned with two files: WORDPR.LSP, which contains the program itself, and TEXTS.LSP, which contains propositional representations for the 14 texts in Table 1. WORDPRO is written in Interlisp-D and will run on any Xerox lisp machine. To use the program, one need only type (Solve name), where name indicates the text to be solved. Three windows are then opened on the computer screen. One displays the propositional representation of the text being solved. The second shows the contents of STM (as in Tables 6 and 7) and the name of the next production rule to be applied. The final window shows the current values of the run-time statistics in Table 8. Before each production is applied, the second and third windows are updated, and the user is prompted to press the left mouse button. This allows the user to step through the program one rule at a time.

Table 8
Run-Time Statistics

Text	No. of Production Rules Fired	No. of Conversions	No. of LTM Searches	Maximum Chunks Held Over
Change-1	15	0	0	2
Change-2	15	0	0	2
Change-3	19	0	0	2
Change-4	19	0	0	2
Change-5	20	1	0	2
Change-6	20	1	0	2
Combine-1	12	0	1	3
Combine-2	12	0	0	2
Compare-1	12	0	1	3
Compare-2	12	0	1	3
Compare-3	13	1	0	2
Compare-4	13	1	0	2
Compare-5	13	1	0	2
Compare-6	13	1	0	2

Note—Columns indicate the number of production rules fired, number of conversions from one schema to another, number of LTM searches, and maximum number of chunks held over from one processing cycle to the next for each problem from Table 1.

REFERENCES

Bovair, S., & Kieras, D. E. (1985). A guide to propositional analysis for research on technical prose. In B. K. Britton & J. B. Black (Eds.), *Understanding expository text* (pp. 315-362). Hillsdale, NJ: Erlbaum.

Fletcher, C. R. (1981). Short-term memory processes in text comprehension. *Journal of Verbal Learning & Verbal Behavior*, **20**, 564-574.

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

Kintsch, W., & van Dijk, T. A. (1978). Toward a model of text comprehension and production. *Psychological Review*, **85**, 363-394.

Newell,A. (1973). Production systems: Models of control structure. In W. G. Chase (Ed.), *Visual information processing* (pp. 463-526). New York: Academic Press.

Turner, A., & Greene, E. (1978). *Construction and use of a propositional text base. JSAS: Catalogue of Selected Documents in Psychology* (MS 1713).

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

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

(Manuscript received February 21, 1985;
revision accepted for publication October 28, 1985.)