Massachusetts Institute of Technology
Cambridge, Massachusetts
Project MAC

Artificial Intelligence Project
Memo 66

Memorandum MAC-M-148
March 30, 1964

NATURAL LANGUAGE INPUT FOR A COMPUTER PROBLEM SOLVING SYSTEM

by Daniel G. Bobrow

Introduction

This paper describes a computer program which accepts and "understands" a comfortable, but restricted set of one natural language, English. Certain difficulties are inherent in this problem of making a machine "understand" English*. Within the limited framework of the subject matter understood by the program, many of these problems are solved or circumvented. I shall describe these problems and my solutions, and point out those solutions which I feel have general applicability. I will also indicate which must be replaced by more general methods to be really useful, and give my ideas about what general solutions to these particular problems might entail.

I shall not bore the reader at this point with a distribe on why one would want to communicate to the computer in English. Suffice it to say that 200 million English speaking people can't be all wrong--and if they could speak to a computer they might even be right more often. Man's ability to use symbols and language is a prime factor in his intelligence, and when we learn how to make a computer understand any natural language, we will have taken a large step toward creating an "artificially intelligent" machine. This is not to say that using "natural language" is necessary; one might do even better to make people change to some more "intelligent" language.

*To avoid excessive circumlocutions, I shall henceforth use just "English" instead of the hedging phrase "restricted subset of English", and use "understand" only in the sense defined below.

The question naturally arises, "What do you mean by having a computer understand natural language?" I have adopted the following operational definition of understanding. A computer understands a subset of English if it will accept input sentences which are members of this subset, and correctly answer questions based on information contained in these sentences. This ability must extend to deductions based on implicit information contained in several sentences. It is desirable that the anwers also be in English to facilitate communication between the computer and a person.

We thus define "understanding" in terms of statements in English. The computer must accept them as input, and answer certain queries about them. How should the computer store the information contained in these statements? If each sentence could be stored unchanged, no information would be lost, but this would put a tremendous burden on the question-answering portion of the program. The question answerer would have to find all relevant sentences, extract the "meaning" pertinent to the question asked, and perform those deductions and manipulations necessary to find the answer to the question asked. For a large corpus, sorting out the relevant material would be a very costly task.

One way of easing the burden of sorting is to create an index for the input corpus. However, unless "meaning" is first extracted from the sentences, the index must be based upon the words in a sentence. The value of the index is then somewhat denigrated by the problems of synonomy and homography.

In a general question-answering system, each type of question may require that meaning be extracted in a different way for convenient manipulation and deduction. Deductive techniques may differ depending on the type of question and the information available in the corpus. To simplify

such problems of sorting for relevant material, extracting meaning, and making inferences, one is driven to select a "point of view", and delimit the type of questions answerable by a system.

One example of a question answering system with a point of view was the SAD SAM program written by Robert Lindsay at Carnegie Tech in 1960. It accepted as input most sentences which could be written in Basic English (a subset of English, designed by C. K. Ogden, which contains a vocabulary of about 1500 words). The questions which it answers are concerned with family relationships between individuals, e.g., "Is Tom the brother of Mary?" or "Who are Jack's grandchildren?" SAD SAM extracts the meaning of a sentence, relevant to family relationship, and stores only that information. Thus from the sentence, "Mary, Tom's sister, went to the meeting," the information about where Mary went would be discarded. The program stores in a family tree type of representation, the information about Mary and Tom's relationship, i.e., it makes them both children of the same (as yet unspecified) pair of parents. The family tree grows as more information is added to the system. To answer a question concerning the relationship

course. An example is:

"The sum of two numbers is 96, and one of the numbers is 16 larger than the other number. Find the two numbers."

Exactly this statement of this problem has been accepted by the STUDENT program and the following solution printed out:

"One of the numbers is 56"

"The other number is 40"

The details of how this is accomplished will be discussed below.

I chose this problem context for a number of reasons. First there is a good form in which to store this type of information for later manipulation, namely as algebraic equations. Secondly, I felt that there was a manageable subset of English in which many of these problems would be expressible, and that this subset could be expanded incrementally. Finally, there are a large number of "algebra story problems" available in first year high school text books.

Since the entire process from input processing to question answering was programmed, a measure of comparison with human performance is available. In fact, the STUDENT program answers most questions that it can handle as fast

the type of information expected. The information storage structure used in STUDENT falls into the category of what I call relational models.

A relational model is defined by three things: a set of objects, the relationships between these objects expressible in the model, and a language media for exhibiting the relationships that exist between particular objects. A relational model is useful for a question answering system if there exist techniques which can take advantage of the relational language to find implied, but not necessarily explicitly stated, relationships between objects of the model.

Lindsay's program for answering questions about family relationships uses such a relational model. The objects in the model are people, and the basic relationship used in the model is the parent-child relationship. The media used for expressing the relationship between individuals is a tree of nodes, with nodes representing individuals, and directed branches representing the parent-child relationship. This model is useful because all other family relationships can be defined in terms of this one basic relationship, and questions about the relationship between two individuals can be answered on the basis of a computation on the path connecting these two individuals in the family tree.

The STUDENT question answering system also uses a relational model. The objects in the model are words and phrases "naming" numbers, or numbers with units attached. I call these objects "variables". The basic relationships are the arithmetic relations of sum, difference, product, quotient, exponentiation and equality. The media for expressing the relationships between objects is a set of equations. The model is useful because well defined techniques exist for finding numerical values which satisfy sets of simultaneous equations. Thus the system can answer questions about the

value of a number named by a given phrase, although this value is given only implicitly in the relationships stated in the problem.

For any relational model used in an "English language" question answering system, there are two important considerations. The first is how can the English input be transformed into the relational language media, and the second is what are good deductive techniques for using the model to solve problems. For algebra story problems a good general format for a relational model was known, based on sets of simultaneous equations. The implementation within the STUDENT program of the transformation and solution procedures, based on this general model, are discussed below.

The Notation Used in STUDENT's Relational Model

The relational model in the STUDENT system uses a set of algebraic equations to represent the arithmetic relationships expressed in the English input. These equations are expressed in a parenthesized prefix notation rather than the conventional infix notation. For example, the conventional infix notation expression, B + C, is written (PLUS B C).

In general, in this prefix notation, the name of the arithmetic function used is made the first element of a list, and succeeding list elements are the variables which are the arguments of that function. The exact notation used is given in Figure 1 below. Note that "minus" is a unary minus, and that the usual binary subtraction operator is a composite relation in the model. In addition, "plus" and "times" are not strictly binary. Indeed, in the model they may have an indefinite number or arguments, e.g., (TIMES,A,B,C) is a legitimate prefix notation expression in the STUDENT model.

Operation	Infix Notation	Prefix Notation
Equality	A = B	(EQUAL A B)
Addition	A + B	(PLUS A B)
Negation	-A	(MINUS A)
Subtraction	A - B	(PLUS A (MINUS B))
Multiplication	A * B	(TIMES A B)
Division	A / B	(QUOTIENT A B)
Exponentiation	A^B	(EXPT A B)

Figure 5

The use of a fully parenthesized notation such as this circumvents the problem of ambiguity in the order in which operations occur. In the expression A + B * C in unparenthesized infix notation, it is unclear whether A is to be added to B and the sum multiplied by C, or if the product of B and C is to be added to A. One solution to this ambiguity is to give each operation a relative precedence, and operations of higher precedence are assumed to be performed first. Such a precedence scheme is assumed by STUDENT in determining an interpretation of similar ambiguous English expressions. Once inside the model, however, arithmetic operations and arguments are fully parenthesized, and therefore, order of operations is unambiguous.

Outline of the Operation of STUDENT

The first step in the operation of the STUDENT question answering system is to provide the STUDENT "program" with some general information which it will "remember" and use, if relevant, in all problems it is asked to do. Through a program called REMEMBER, this information becomes part of the permanent store of knowledge in the system and is what I call global information. Some examples of global facts which have been given to the STUDENT system are:

"12 inches equals 1 foot"

or

"twice always means 2 times"

How these facts are used will be described later.

Then STUDENT is asked to solve a particular problem. To do this, it transforms the English statement of the problem into the media of the relational model, and then manipulates objects in the model to find the answer. More specifically, STUDENT transforms the English input into a set of simultaneous equations, keeping a list of what answers are required, a list of the units involved (e.g., dollars, pounds) and a list of all the variables in the equations. Then STUDENT invokes the SOLVE routine to solve this set of equations for the desired unknowns. If a solution is found, STUDENT prints the values of the unknowns requested in the format illustrated earlier, i.e., substituting in "(variable is value)" the appropriate phrases for variable and value. If a solution cannot be found, various heuristics are used to identify two variables (i.e., find two slightly different phrases that refer to the same number). If two variables, A and B, are identified the equation A = B is added to the set of equations. Reference is made to the store of global information to find any relevant equations.

Assumptions made about the identity of variables, and the possibly relevant stored equations retrieved by STUDENT are printed out. If use of these identification or global equations leads to a solution, then the result is printed out in the format described above.

If a solution was not found, and certain idioms are present in the English statement of the problem, then a substitution is made for each of these idioms in turn, and the transformation and solution processes are

repeated. If it is unsuccessful with all single substitutions STUDENT reports the failure and terminates. If the problem is ever solved, the solution is printed and the program terminates.

Transformation of the English Input to the Relational Model

The words and phrases (strings of words) in the English input can be classified into three distinct categories on the basis of how they are handled in the transformation. The first category consists of strings of words which denote objects in the model; I call such strings, variables. Variables are identified only by the string of words in them, and if two strings differ at all, they define distinct variables. One important problem considered below is how to determine when two distinct variables refer to the same object.

The second class of words and phrases are what I call "substitutors". Each substitutor may be replaced by another string. Some substitutions are mandatory; others are optional and are only made if the problem cannot be solved without such substitutions. An example of a mandatory substitution is "2 times" for the word "twice". "Twice" always means "2 times" in the context of the model, and therefore this substitution is always made. One optional "idiomatic" substitution is "twice the sum of the length and width of the rectangle" for "the perimeter of the rectangle". The use of these substitutions in the transformation process is discussed below.

Members of the third class of words and phrases indicate the relationships between the objects in the model, i.e., the variables in the problem. I call members of this third class "operators". Operators may indicate operations which are complex combinations of the basic relationships. One simple operator is the word "plus", which indicates the operation of addition

A complex operator is the phrase "percent less than", as in "10 percent less than the marked price", which locates the number immediately preceding "percent", subtracts it from 100, divides this result by 100, and then multiplies this quotient by the variable following the "than".

The class of operators may be further subdivided according to where the arguments of the operators are found. A prefix operator, such as "the square of..." precedes its argument. An operator like "... percent" is a suffix operator, and follows its argument. Infix operators such as "... plus ..." or "... less than ..." appear between their two arguments. In a split prefix operator such as "difference between ... and ...", part of the operator precedes, and part appears between the two arguments. "The sum of ... and ... and ..." is a split prefix operator with an indefinite number of arguments.

Some words may conditionally act as operators, depending on their context. For example, "of" is only equivalent to "times" if there is a number immediately preceding it; e.g., ".5 of the profit" is equivalent to ".5 times the profit"; however, "Queen of England" does not imply a multiplicative relationship between the Queen and her country.

Let us now consider in detail the transformation procedure used by STUDENT and see how these different types of phrases interact. To make the process more concrete, let us consider the following example which has been solved by STUDENT.

(THE PROBLEM TO BE SOLVED IS)
(IF THE NUMBER OF CUSTOMERS TOM GETS IS TWICE THE SQUARE OF 20 PER CENT OF THE NUMBER OF ADVERTISEMENTS HE RUNS, AND THE NUMBER OF ADVERTISEMENTS HE RUNS IS 45, WHAT IS THE NUMBER OF CUSTOMERS TOM GETS Q.)

Figure 2

This text is a copy of actual printout from the program, showing stages in the transformation and the solution of the problem. The parentheses are an artifact of the LISP programming language, and "Q.", is a replacement for the question mark not available on the key punch.

The first stage in the transformation is to perform all mandatory substitutions. In this problem, only the three phrases underlined (single words are one word phrases) are substitutors: "twice" becomes "2 times", "per cent" becomes the single word "percent", and "square of" is truncated to "square". Having made these substitutions, STUDENT prints:

(WITH MANDATORY SUBSTITUTIONS THE PROBLEM IS)
(IF THE NUMBER OF CUSTOMERS TOM GETS IS 2 TIMES THE SQUARE 20 PERCENT OF THE NUMBER OF ADVERTISEMENTS HE RUNS, AND THE NUMBER OF ADVERTISEMENTS HE RUNS IS 45, WHAT IS THE NUMBER OF CUSTOMERS TOM GETS Q.)

Figure 3

Using dictionary entries for each word, the words in the problem are now tagged by their function in terms of the transformation process, and STUDENT prints:

(WITH WORDS TAGGED BY FUNCTION THE PROBLEM IS)
(IF THE NUMBER (OF / OP) CUSTOMERS TOM (GETS / VERB) IS 2 (TIMES / OP 1) THE (SQUARE / OP 1) 20 (PERCENT / OP 2) (OF / OP) THE NUMBER (OF / OP) ADVERTISEMENTS (HE / PRO) RUNS, AND THE NUMBER (OF / OP) ADVERTISEMENTS (HE / PRO) RUNS IS 45, (WHAT / QWORD) IS THE NUMBER (OF / OP) CUSTOMERS TOM (GETS / VERB) (QMARK / DLM))

Figure 4

If a word has a tag, or tags, the word followed by "/", followed by the tags,

becomes a single unit, and is enclosed in parentheses. Typical taggings are indicated in Figure 4. "(OF / OF)" indicates that "of" is an operator and other taggings show that "gets" is a verb, "times" is an operator of level 1 (operators levels will be explained below), "square" is an operator of level 1, "percent" is an operator of level 2, "he" is a pronoun, "what" is a question word, and "QMARK" (replacing Q.) is a delimiter of a sentence. These tagged words will play the principal role in the remaining transformation to the set of equations implicit in this problem statement.

The next stage in the transformation is to break the input sentences into "simple sentences". As in the example, a problem may be stated using sentences of great grammatical complexity; but the final stage of the transformation is only defined on a set of simple sentences. The method adopted here to perform this analysis is ad hoc and primitive, but works reasonably well because of the limited number of ways in which algebra story problems are expressed. This problem of extraction of simple understandable sentences occurs in any general language processor.

The simplification method employed in STUDENT depends on the recursive use of format matching. If an input sentence is of the form "if" followed by a substring, followed by a comma, a question word and a second substring (i.e., matches the COMIT left half IF + $ + , + $1/QWORD + $ =) then the first substring (between the IF and the comma) is made an independent sentence, and everything following the comma is made a second sentence. In the example, this means that the input is resolved into the two sentences, (where tags are omitted for the sake of brevity):

"The number of customers Tom gets is 2 times the square 20 percent of

the number of advertisements he runs, and the number of advertisements he runs is 45." and "What is the number of customers Tom gets?"

This last procedure effectively resolves a problem into declarative assumptions and a question sentence. A second complexity resolved by STUDENT is illustrated in the first sentence of this pair. A coordinate sentence consisting of two sentences joined by a comma immediately followed by an "and" (i.e., any sentence matching the COMMA left half $ + , + AND + $) will be resolved into the two independent sentences. The first sentence above is therefore resolved into two simpler sentences.

Using these two ad hoc format simplifications, the problem statement is put into canonically "simple" sentences. For the example, STUDENT prints

```
(THE SIMPLE SENTENCES ARE)

(THE NUMBER (OF / OP) CUSTOMERS TOM (GETS / VERB) IS 2 (TIMES / OP 1)
THE (SQUARE / OP 1) 20 (PERCENT / OP 2) (OF / OP) THE NUMBER (OF / OP)
ADVERTISEMENTS (HE / PRO) RUNS (PERIOD / DLM))

(THE NUMBER (OF / OP) ADVERTISEMENTS (HE / PRO) RUNS IS 45
(PERIOD / DLM))

((WHAT / QWORD) IS THE NUMBER (OF / OP) CUSTOMERS TOM (GETS / VERB)
(QMARK / DLM))
```

Figure 5

Each simple sentence is a separate list, i.e., is enclosed in parentheses, and each ends with a delimiter (a period or question mark). Each of these sentences can now be transformed directly to its interpretation in the model.

nized by the STUDENT program. New operators can easily be added to the program equivalent of this table.

In performing the transformation of a phrase P, a left to right search is made for an operator of level 2 (indicated by subscripts of "OP" and 2). If none is found, a left to right search is made for a level 1 operator (indicated by subscripts "OP" and 1), and finally another left to right search for an operator of level 0 (indicated by a subscript "OP" and no numerical subscript). If an operator is found, this operator and its context are transformed as indicated in column 4 in the table. If no operator is present, delimiters and articles (a, an and the) are deleted and the phrase is treated as an indivisible entity, a variable.

In the example, the first simple sentence is

(THE NUMBER (OF/OP) CUSTOMERS TOM (GETS/VERB) IS 2 (TIMES/OP 1) THE (SQUARE/OP 1) 20 (PERCENT/OP 2) (OF/OP) THE NUMBER (OF/OP) ADVERTISEMENTS (HE/PRO) RUNS (PERIOD/DLM))

This is of the form "P1 is P2", and is transformed to (EQUAL P1* P2*). P1 is "(THE NUMBER (OF/OP) CUSTOMERS TOM (GETS/VERB))". The occurrence of the verb "gets" is ignored because of the presence of the "is" in the sentence, meaning "equals". The only operator found is "(OF/OP)". From the table we see that if "of" is immediately preceded by a number (not the word "number") it is treated as if it were the infix "times". In this case, however, "of" is not preceded by a number, the subscript OP indicating that "of" is an operator is stripped away, and the transformation process is repeated on the phrase with "of" no longer acting as an operator. In this repetition, no operators are found, and P1* is the variable

(NUMBER OF CUSTOMERS TOM (GETS/VERB)).

To the right of "is" in the sentence is P2:

(2 (TIMES/OP 1) THE (SQUARE/OP 1) 20 (PERCENT/OP 2) (OF/OP) THE NUMBER

(OF/OP) ADVERTISEMENTS (HE/PRO) RUNS (PERIOD/DIM))

Operators in STUDENT

Operator	Precedence Level	Context	Transformation to Interpretation in the Model (a)	
PLUS	2	P1 PLUS P2	(PLUS,P1*,P2*)	
PLUSS	0	P1 PLUSS P2	(PLUS,P1*,P2*)	
MINUS	2	P1 MINUS P2	(PLUS,P1*,(MINUS P2*))	(b)
		MINUS P2	(MINUS P2*)	
MINUSS	0	P1 MINUSS P2	(PLUS P1*,(MINUS P2*))	
TIMES	1	P1 TIMES P2	(TIMES P1* P2*)	
DIVBY	1	P1 DIVBY P2	(QUOTIENT P1* P2*)	
SQUARE	1	SQUARE P1	(EXPT P1* 2)	(c)
SQUARED	0	P1 SQUARED	(EXPT P1* 2)	
**	0	P1 ** P2	(EXPT P1* P2*)	
LESSTHAN	2	P1 LESSTHAN P2	(PLUS P2* (MINUS P1*))	
PER	0	P1 PER K P2 P1 PER P2	(QUOTIENT P1* (K P2)*) (QUOTIENT P1* (1 P2)*)	(d)
PERCENT	2	P1 K PERCENT P2	(P1(K/100) P2)*	(f)
PERLESS	2	P1 K PERLESS P2	(P1((100-K)/100) P2)*	(f)
SUM	0	SUM P1 AND P2 AND P3	(PLUS P1* (SUM P2 AND P3)*)	(c)
		SUM P1 AND P2	(PLUS P1* P2*)	
DIFFERENCE	0	DIFFERENCE BETWEEN P1 AND P2	(PLUS P1* (MINUS P2*))	
OF	0	K OF P2	(TIMES K P2*)	
		P1 OF P2	(P1 OF P2)*	

(a) If P1 is a phrase, P1* indicates its interpretation in the model.

(b) When two possible contexts are indicated, they are checked in the order shown.

(c) SQUARE P1 and SUM P1 are idiomatic shortenings of SQUARE OF P1 and SUM OF P1.

(d) * outside a parenthesized expression indicates that the entire phrase enclosed is to be transformed.

(e) K is a number.

(f) / and - imply that the indicated arithmetic operations are actually performed.

The first operator found in P2 is PERCENT, an operator at level 2. From the table in Figure 6, we see that this operator has the effect of dividing the number immediately preceding it by 100. The "PERCENT" is removed and the transformation is repeated on the remaining phrase. In the example, the "...20 (PERCENT/OP 2) (OF/OP) ..." becomes "... .2000 (OF/OP) ...".

Continuing the transformation, the operators found are, in order, TIMES, SQUARE, OF and OF. Each is handled as indicated in the table. The "of" in the context "... .2000 (OF/OP) THE ..." is treated an infix TIMES, while at the other occurrence of "OF", the operator marking is removed. The resulting transformed expression for P2 is:

(TIMES 2 (EXPT (TIMES .2 (NUMBER OF ADVERTISEMENTS (HE/PRO) RUNS)) 2))

The transformation of the second sentence of the example is done in a similar manner, and yields the equation:

(EQUAL (NUMBER OF ADVERTISEMENTS (HE/PRO) RUNS) 45)

The third sentence is of the form "What is P1?". It starts with a question word and is therefore treated specially. A unique variable, a single word consisting of an X followed by five integers, is created, and the equation (EQUAL Xnnnnn P1*) is stored. For this example, the variable X00001 was created, and this last simple sentence is transformed to the equation:

(EQUAL X00001 (NUMBER OF CUSTOMERS TOM (GETS/VERB))

In addition, the created variable is placed on the list of variables for which STUDENT is to find a value. Also, this variable is stored, paired with P1, the untransformed right side, for use in printing out the answer. If a value is found for this variable, STUDENT prints the sentence (P1 is

value) with the appropriate substitution for value. Figure 7 shows the full set of equations, and the printed solution given by STUDENT for the example being considered. For ease in solution, the last equations created are the first in this list of equations.

```
(THE EQUATIONS TO BE SOLVED ARE)

(EQUAL X00001 (NUMBER OF CUSTOMERS TOM (GETS / VERB)))

(EQUAL (NUMBER OF ADVERTISEMENTS (HE / PRO) RUNS) 45)

(EQUAL (NUMBER OF CUSTOMERS TOM (GETS / VERB)) (TIMES 2 (EXPT
(TIMES .2000 (NUMBER OF ADVERTISEMENTS (HE / PRO) RUNS)) 2)))

(THE NUMBER OF CUSTOMERS TOM GETS IS 162)
```

Figure 7

In the example just shown, the equality relation was indicated by the copula "is". In the problem solved by STUDENT shown in Figure 8 below, equality is indicated by the occurrence of a transitive verb in the proper context.

```
(THE PROBLEM TO BE SOLVED IS)
(TOM HAS TWICE AS MANY FISH AS MARY HAS GUPPIES . IF MARY HAS
3 GUPPIES , WHAT IS THE NUMBER OF FISH TOM HAS Q.)

(THE EQUATIONS TO BE SOLVED ARE)

(EQUAL X00001 (NUMBER OF FISH TOM (HAS / VERB)))

(EQUAL (NUMBER OF GUPPIES (MARY / PERSON) (HAS / VERB)) 3)

(EQUAL (NUMBER OF FISH TOM (HAS / VERB)) (TIMES 2 (NUMBER OF
GUPPIES (MARY / PERSON) (HAS / VERB))))

(THE NUMBER OF FISH TOM HAS IS 6)
```

Figure 8

The verb in this case is "has". The simple sentence "Mary has 3 guppies" is transformed to the "equivalent" sentence "The number of guppies Mary

has is 3" and the processing of this latter sentence is done as previously discussed.

The general format for this type of sentence, and the format of the intermediate sentence to which it is transformed is best expressed by the following COMIT transformation rule:

$ + $1/VERB + $1/NUMBER + $ = THE + NUMBER + OF + 4 + 1 + 2 + IS + 3

This may be read as--anything (a subject) followed by a verb followed by a number followed by anything (the unit) is transformed to a sentence starting with "THE NUMBER OF" followed by the unit, followed by the subject and the verb, followed by "IS" and then the number. In "Mary has 3 guppies" the subject is "Mary", the verb "has", and the units "guppies". Similarly, the sentence "The witches of Firth brew 3 magic potions" would be transformed to

"The number of magic potions the witches of Firth brew is 3."

In addition to a declaration of number, a single object transitive verbs may be used in a comparative structure, such as exhibited in the sentence "Tom has twice as many fish as Mary has guppies." The COMIT rule which gives the effective transformation for this type of sentence structure is:

$ -- $1/VERB + $ + AS + MANY + $ + AS + $ + $1/VERB + $ =

THE + NUMBER + OF + 6 + 1 + 2 + IS + 3 + THE + NUMBER + OF

+ 10 + 8 + 9

For the example, the transformed sentence is:

"The number of fish Tom has is twice the number of guppies Mary has."

Transformation of new sentence formats to formats previously "understood" by the program can be easily added to the program, thus extending the subset of English "understood" by STUDENT. In the processing that

actually takes place within STUDENT the intermediate sentence never exists. It is easier to go directly to the model from the format, utilizing subroutines previously defined in terms of the semantics of the model.

The word "is" indicates equality only if it is not used as an auxiliary. The example in Figure 9 shows how verbal phrases containing "is", such as "is multiplied by", and "is increased by" are handled in the transformation.

```
(THE PROBLEM TO BE SOLVED IS)
(A NUMBER IS MULTIPLIED BY 6 . THIS PRODUCT IS INCREASED BY 44 .
THIS RESULT IS 68 . FIND THE NUMBER .)

(THE EQUATIONS TO BE SOLVED ARE)

(EQUAL X00001 (NUMBER))

(EQUAL (PLUS (TIMES (NUMBER) 6) 44) 68)

(THE NUMBER IS 4)
```

Figure 9

The sentence "A number is multiplied by 6" only indicates that two objects in the model are related multiplicatively, and does not indicate explicitly any equality relation. The interpretation of this sentence in the model is the prefix notation product:

(TIMES (NUMBER) 6)

This latter phrase is stored in a temporary location for possible later reference. In this problem, it is referenced in the next sentence, with the phrase "THIS PRODUCT". The important word in this last phrase is "THIS"--STUDENT ignores all words in a variable containing the key word "THIS". The last temporarily stored phrase is substituted for the "this" variable. Thus, the first three sentences in the problem stated in Figure 9 yield only one equation, after two substitutions for "this" phrases. The

last sentence "Find the number." is transformed as if it were "What is the number Q.", and yields the first equation shown.

The word "this" may occur in a context where it is not referring to a previously stored phrase. In Figure 10 is an example of such a context.

```
(THE PROBLEM TO BE SOLVED IS)
(THE PRICE OF A RADIO IS 69.70 DOLLARS . IF THIS PRICE IS
15 PERCENT LESS THAN THE MARKED PRICE , FIND THE MARKED PRICE
.)

(THE EQUATIONS TO BE SOLVED ARE)

(EQUAL X00001 (MARKED PRICE))

(EQUAL (PRICE OF RADIO) (TIMES .8499 (MARKED PRICE)))

(EQUAL (PRICE OF RADIO) (TIMES 69.70 (DOLLARS)))

(THE MARKED PRICE IS 82 DOLLARS)
```

Figure 10

In such contexts, the phrase containing "this" is replaced by the left-half of the last equation created. In the example, the phrase "this price" is replaced by "the price of a radio".

The problem in Figure 10 illustrates two other features of the STUDENT program. The first is the action of the complex operator "percent less than". It causes the number immediately preceding it, i.e., 15, to be subtracted from 100, this result divided by 100, to give .85 (.8499 due to rounding errors in conversion). Then this operator becomes the infix operator "TIMES". This is as indicated in the table in Figure 6.

This problem also illustrates how units such as "dollars" are handled by the STUDENT program. Any word which immediately follows a number is labelled as a special type of variable called a unit. A number followed by a unit is treated in the equation as a product of the number and the unit, e.g., "(TIMES 69.70 (DOLLARS))". Units are treated specially in solving

the set of equations, in that any unit may appear in the answer. If the value for a variable found by the solver is the product of a number and a unit, STUDENT concatenates the number and unit. For example, the solution for "(MARKED PRICE)" in the problem in Figure 10 was (TIMES 82 (DOLLARS)) and STUDENT prints out

"(THE MARKED PRICE IS 82 DOLLARS)

There is an exception to the fact that any unit may appear in the answer, as illustrated in Figure 11.

```
(THE PROBLEM TO BE SOLVED IS)
(IF 1 SPAN EQUALS 9 INCHES , AND 1 FATHOM EQUALS 6 FEET , HOW
MANY SPANS EQUALS 1 FATHOM Q.)

(THE EQUATIONS TO BE SOLVED ARE)

(EQUAL X00001 (TIMES 1 (FATHOMS)))

(EQUAL (TIMES 1 (FATHOMS)) (TIMES 6 (FEET)))

(EQUAL (TIMES 1 (SPANS)) (TIMES 9 (INCHES)))

UNABLE TO SOLVE THIS SET OF EQUATIONS

(USING THE FOLLOWING KNOWN RELATIONSHIPS)
((EQUAL (TIMES 1 (YARDS)) (TIMES 3 (FEET))) (EQUAL (TIMES 1
(FEET)) (TIMES 12 (INCHES))))

(1 FATHOM IS 8 SPANS)
```

Figure 11

If, as in the problem in Figure 11, the unit of the answer is specified--by the phrase "how many spans"--then only that unit, in this case spans, may appear in the answer. Without this restriction, STUDENT would blithely answer this problem with "(1 FATHOM IS 1 FATHOM)".

In the transformation from the English statement of the problem to the equations, 9 inches become (TIMES 9 (INCHES)). However, 1 fathom became (TIMES 1 (FATHOMS)). The plural form for fathom has been substi-

tuted for the singular form. STUDENT always uses the plural form if known, to ensure that all units appear in only one form. Since "fathom" and "fathoms" are different STUDENT would treat them as distinct, unrelated units. The plural form is part of the global information that can be made available to STUDENT and the plural form of a word is substituted for any singular form appearing after "1" in any phrase. The inverse operation is carried out to perform correct printout of the solution.

Notice that the information given in the problem was insufficient to allow solution to the set of equations to be solved. Therefore, STUDENT looked in its glossary for information concerning each of the units in this set of equations. It found the relationship "1 foot equals 12 inches". Using this fact, and the equation it implies, STUDENT is able to solve the problem. Thus, in certain cases where a problem is not "analytic", in the sense that it does not contain, explicitly stated, all the information needed for its solution, STUDENT is able to draw on a body of facts, pick out relevant ones, and use these to obtain a solution.

There is another class of problems which I call semi-analytic. In such problems, the transformation process does not yield a set of solvable equations. However, in this set of equations there exists a pair of variables (or more than one pair) such that the two variables are only "slightly" different, and really name the same object in the model. When a set of equations is unsolvable STUDENT searches for relevant global equations. In addition, it uses several heuristic techniques for identifying two slightly different variables in the equations. The problem in Figure 11 illustrates identification of two variables where in one variable a pronoun has been substituted for a noun phrase in the other variable. This identification is made by checking all variables appearing before one containing the pro-

noun, and finding one which is identical to this pronoun phrase, with a

```
(THE PROBLEM TO BE SOLVED IS)
(THE NUMBER OF SOLDIERS THE RUSSIANS HAVE IS ONE HALF OF THE
NUMBER OF GUNS THEY HAVE . THE NUMBER OF GUNS THEY HAVE IS
7000 . WHAT IS THE NUMBER OF SOLDIERS THEY HAVE Q.)

(THE EQUATIONS TO BE SOLVED ARE)

(EQUAL X00001 (NUMBER OF SOLDIERS (THEY / PRO) (HAVE / VERB)))

(EQUAL (NUMBER OF GUNS (THEY / PRO) (HAVE / VERB)) 7000)

(EQUAL (NUMBER OF SOLDIERS RUSSIANS (HAVE / VERB)) (TIMES .5000
(NUMBER OF GUNS (THEY / PRO) (HAVE / VERB))))

UNABLE TO SOLVE THIS SET OF EQUATIONS

(ASSUMING THAT)
((NUMBER OF SOLDIERS (THEY / PRO) (HAVE / VERB)) IS EQUAL TO
(NUMBER OF SOLDIERS RUSSIANS (HAVE / VERB)))

(THE NUMBER OF SOLDIERS THEY HAVE IS 3500)
```

Figure 12

substitution of a string any length for the pronoun. If two variables match in this fashion, STUDENT assumes the two variables are equal, and prints out a statement of this assumption, as shown. The solution procedure is then tried again, with the additional equations from identifications. In the example, the additional equation was sufficient to determine the solution.

The example in Figure 13 is again a "non-analytic" problem. The first set of equations developed by STUDENT is unsolvable. Therefore, STUDENT tries to find some relevant equations in its store of global information. It uses the first word of each variable string as a key to its glossary. The one exception to this rule is that the words "number of" are ignored if they are the first two words of a variable string. Thus, in this problem, STUDENT retrieved equations which were stored under the

key words distance, gallons, gas, and miles. Two facts about distance had been stored earlier; "distance equals speed times time" and "distance equals gas consumption times number of gallons of gas used". The equations implicit in these sentences were stored and retrieved now--as possibly relevant to the solution. In fact, only the second is relevant.

```
(THE PROBLEM TO BE SOLVED IS)
(THE GAS CONSUMPTION OF MY CAR IS 15 MILES PER GALLON . THE
DISTANCE BETWEEN BOSTON AND NEW YORK IS 250 MILES . WHAT IS
THE NUMBER OF GALLONS OF GAS USED ON A TRIP BETWEEN NEW YORK
AND BOSTON Q.)

(THE EQUATIONS TO BE SOLVED ARE)

(EQUAL X00001 (NUMBER OF GALLONS OF GAS USED ON TRIP BETWEEN
NEW YORK AND BOSTON))

(EQUAL (DISTANCE BETWEEN BOSTON AND NEW YORK) (TIMES 250 (MILES)))

(EQUAL (GAS CONSUMPTION OF MY CAR) (QUOTIENT (TIMES 15 (MILES))
(TIMES 1 (GALLONS))))

UNABLE TO SOLVE THIS SET OF EQUATIONS

(USING THE FOLLOWING KNOWN RELATIONSHIPS)
((EQUAL (DISTANCE) (TIMES (SPEED) (TIME))) (EQUAL (DISTANCE)
(TIMES (GAS CONSUMPTION) (NUMBER OF GALLONS OF GAS USED))))

(ASSUMING THAT)
((DISTANCE) IS EQUAL TO (DISTANCE BETWEEN BOSTON AND NEW YORK))

(ASSUMING THAT)
((GAS CONSUMPTION) IS EQUAL TO (GAS CONSUMPTION OF MY CAR))

(ASSUMING THAT)
((NUMBER OF GALLONS OF GAS USED) IS EQUAL TO (NUMBER OF GALLONS
OF GAS USED ON TRIP BETWEEN NEW YORK AND BOSTON))

(THE NUMBER OF GALLONS OF GAS USED ON A TRIP BETWEEN NEW YORK
AND BOSTON IS 16.66 GALLONS)
```

Figure 13

Before any attempt is made to solve this augmented set of equations, the variables of the augmented set are matched, to identify "slightly different" variables which refer to the same object in the model. In this example "(DISTANCE)", "(GAS CONSUMPTION)" and "(NUMBER OF GALLONS OF GAS USED)", are all identified with "similar" variables. The following heuristically determined conditions must be satisfied for identification of variables P1 and P2.

1) P1 must appear later in the problem than P2.

2) P1 is completely contained in P2 in the sense that P1 is a contiguous substring within P2.

This identification reflects a syntactic phenomenon where a truncated phrase, with one or more modifying phrases dropped, is often used in place of the entire phrase. For example, if the phrase "the length of a rectangle" has occurred, the phrase "the length" may be used to mean the same thing. This identification is distinct from that made using pronoun substitution.

In the example in Figure 13, a schema is used by identifying the variables in the schema with the variables that occur in the problem. This problem is solvable exactly because the key phrases "distance", "gas consumption" and "number of gallons of gas used" occur as substrings of the variables in the problem. Since STUDENT identifies each generic key phrase of the schema with a particular variable of the problem, any schema can be used only once in a problem. Because STUDENT handles schema in this ad hoc fashion it cannot solve problems in which a relstionship such as "distance equals speed times time" is needed for two different values of distance, speed, and time. With some effort, this weakness in the program could be overcome.

```
(THE PROBLEM TO BE SOLVED IS)
(THE LENGTH OF A RECTANGLE IS 8 INCHES MORE THAN THE WIDTH
OF THE RECTANGLE . ONE HALF OF THE PERIMETER OF THE RECTANGLE
IS 18 INCHES . FIND THE LENGTH AND THE WIDTH OF THE RECTANGLE
.)

(THE EQUATIONS TO BE SOLVED ARE)

(EQUAL X00001 (WIDTH OF RECTANGLE))

(EQUAL X00002 (LENGTH))

(EQUAL (TIMES .5000 (PERIMETER OF RECTANGLE)) (TIMES 18 (INCHES)))

(EQUAL (LENGTH OF RECTANGLE) (PLUS (TIMES 8 (INCHES)) (WIDTH
OF RECTANGLE)))

UNABLE TO SOLVE THIS SET OF EQUATIONS

TRYING POSSIBLE IDIOMS

(THE PROBLEM WITH AN IDIOMATIC SUBSTITUTION IS)
(THE LENGTH OF A RECTANGLE IS 8 INCHES MORE THAN THE WIDTH
OF THE RECTANGLE . (ONE HALF OF TWICE THE SUM OF THE LENGTH
AND WIDTH OF THE RECTANGLE IS 18 INCHES . FIND THE LENGTH AND
THE WIDTH OF THE RECTANGLE .)

(THE EQUATIONS TO BE SOLVED ARE)

(EQUAL X00003 (WIDTH OF RECTANGLE))

(EQUAL X00004 (LENGTH))

(EQUAL (TIMES (TIMES .5000 2) (PLUS (LENGTH) (WIDTH OF RECTANGLE)))

(TIMES 18 (INCHES)))

(EQUAL (LENGTH OF RECTANGLE) (PLUS (TIMES 8 (INCHES)) (WIDTH
OF RECTANGLE)))

UNABLE TO SOLVE THIS SET OF EQUATIONS

(USING THE FOLLOWING KNOWN RELATIONSHIPS)
((EQUAL (TIMES 1 (FEET)) (TIMES 12 (INCHES))))

(ASSUMING THAT)
((LENGTH) IS EQUAL TO (LENGTH OF RECTANGLE))

(THE LENGTH IS 13 INCHES)

(THE WIDTH OF THE RECTANGLE IS 5 INCHES)
```

Figure 15

Special Heuristics

The methods thus far discussed have been applicable to the entire range of algebra problems. However, for special classes of problems, additional heuristics may be used which are needed for members of the class, but not applicable to other problems. An example is the class of age problems, as typified by the problem in Figure 16.

Before the age problem heuristics are used, a problem must be identified as belonging to that class. STUDENT identifies age problems by any occurrence of one of the following phrases, "as old as", "years old" and "age". This identification is made immediately after all words are looked up in the dictionary and tagged by function. After the special heuristics are used the modified problem is transformed to equations exactly as stated previously.

```
(THE PROBLEM TO BE SOLVED IS)
(BILL S FATHER S UNCLE IS TWICE AS OLD AS BILL S FATHER . 2
YEARS FROM NOW BILL S FATHER WILL BE 3 TIMES AS OLD AS BILL
. THE SUM OF THEIR AGES IS 92 . FIND BILL S AGE .)

(THE EQUATIONS TO BE SOLVED ARE)

(EQUAL X0001 ((BILL / PERSON) S AGE))

(EQUAL (PLUS ((BILL / PERSON) S (FATHER / PERSON) S (UNCLE
/ PERSON) S AGE) (PLUS ((BILL / PERSON) S (FATHER / PERSON)
S AGE) ((BILL / PERSON) S AGE))) 92)

(EQUAL (PLUS ((BILL / PERSON) S (FATHER / PERSON) S AGE) 2)
(TIMES 3 (PLUS ((BILL / PERSON) S AGE)))

(BILL S AGE IS 8)
```

Figure 16

The need for special methods for age problems arises because of the conventions used for denoting the variables, all of which are ages. The word age is usually not used explicitly, but is implicit in such phrases as "as old as". People's names are used where their ages are really the

implicit variables. In the example, for instance, the phrase "Bill's father's uncle" is used to refer to "Bill's father's uncle's age".

STUDENT uses a special heuristic to make all these ages explicit. To do this, it must know which words are "person words" and therefore, may be associated with an age. For this problem STUDENT has been told that Bill, father, and uncle are person words. They can be seen tagged as such in the equations. The "s" following a word is the STUDENT representation for possessive, used instead of "apostrophe - s" for programming convenience. STUDENT inserts a "S AGE" after every person word not followed by a "S" (because this "S" indicates that the person word is being used in a possessive sense, not as an independent age variable). Thus, as indicated, the phrase "BILL S FATHER S UNCLE" becomes "BILL S FATHER S UNCLE S AGE".

In addition to changing phrases naming people to ones naming ages, STUDENT makes certain special idiomatic substitutions. For the phrase "their ages", STUDENT substitutes all the age variables encountered in the problem. In the example, for "THEIR AGES" STUDENT substitutes "BILL S FATHER S UNCLE S AGE AND BILL S FATHER S AGE AND BILL S AGE". The phrases "as old as" and "years old" are then deleted as dummy phrases not having any meaning, and "will be" and "was" are changed to "IS". There is no need to preserve the tense of the copula, since the sense of the future tense is preserved in such prefix phrases as "2 years from now".

The remaining special age problem heuristics are used to process the phrases "in 2 years", "5 years ago" and "now". The phrase "2 years from now" is transformed to "in 2 years" before processing. These three time phrases may occur immediately after the word age, (e.g., "Bill's age 3 years ago") or at the beginning of the sentence. If a time phrase occurs at the at the beginning of the sentence, it implicitly modifies all ages mentioned

sentence becomes the two sentences: "Mary is twice as old as Ann X00007 years ago. X00007 years ago Mary was as old as Ann is now." These two occurrences of time phrases are handled as discussed previously. Similarly the phrase "will be when" would be transformed to "in K years . In K years".

These decoupling heuristics are useful not only for the STUDENT program but for people trying to solve age problems. The classic age problem in Figure 17 took an MIT graduate student over 5 minutes to solve because he did not know this heuristic. With the heuristic he was able to set up the appropriate equations much more rapidly. As a crude measure of STUDENT's relative skill, note that STUDENT took less than one minute to solve the problem in Figure 17.

Global Information

This algebra problem-solving system contains two programs which process English input. One is the program thus far discussed, STUDENT, which accepts the statement of an algebra story problem and attempts to find the solution to the particular problem. STUDENT does not store any information, nor "remember" anything from problem to problem.

The other program is called REMEMBER and it processes and stores facts not specific to any one problem. These facts make up STUDENT's store of "global information". This information is accepted in a subset of English which overlaps but is different from the subset of English accepted by STUDENT. REMEMBER accepts statements in certain fixed formats. The following are the formats currently understood, and the method of storage of the information obtained in each format.

1) Example: Distance equals speed times time.

Format: P1 equals P2.

Processing: The sentence is transformed into an equation in the same way it is done in STUDENT. This equation is stored on the property lists of the atoms which are the first words in each variable. In the example, the equation

"(EQUAL (DISTANCE) (TIMES (SPEED) (TIME)))"

is stored on the property list of "DISTANCE", "SPEED" and "TIME". This equation will be retrieved if needed and one of these words appears in the problem.

2) Example: Times is an operator of level 1.

Format: P1 is an operator of level K.

Processing: A dictionary entry for P1 is created with subscripts of OP and K. For TIMES, the dictionary entry (TIMES/OP 1) is created. These entries are those that are used to determine the tagging of words by function.

3) Example: OF is an operator.

Format: P1 is an operator.

Processing: Creates a dictionary entry for P1 with the subscript OP. The entry for OF is (OF/OP).

4) Example: Bill is a person.

Format: P1 is a P2.

Processing: Creates a dictionary entry for P1, with P2 as a subscript. The entry for BILL is (BILL/PERSON).

5) Example: Feet is the plural of foot.

Format: P1 is the plural of P2.

Processing: On the property list of P1, after the flag SING, P2 is stored; on the property list of P2, after the flag PLURAL, the word P1 is stored. Thus FEET is stored after PLURAL on the property list of the

atom FOOT.

6) Example: One half always means 0.5.

Format: P1 always means P2.

Processing: The program STUDENT is actually modified so that if P1 occurs, the mandatory substitution of P2 for P1 will be made. The last sentence of this format processed by REMEMBER will be the first mandatory substitution made. Thus "one always means 1" followed by "one half always means 0.5" will cause the desired substitutions to be made; if these sentences were reversed no occurrence of "one half" would ever be found since it would have been charged to "1 half".

7) Example: Two numbers sometimes means one number and the other number.

Format: P1 sometimes means P2.

Processing: The STUDENT program is modified so that the possible idiomatic substitution of P2 for P1 will be made in a problem if it is otherwise unsolvable. All such "possible idiomatic substitutions" are tried when necessary, with the last one entered being the first one tried.

These last two formats actually insert new METEOR program statements into STUDENT. If P1 and P2 are METEOR left and right halves respectively, using special METEOR features, these formats can be used to extend the subset of English understood by STUDENT.

Conclusion

The STUDENT program accepts as input an algebra story problem couched in a restricted subset of English. It tries to solve this problem by mapping the input into a relational model and manipulating structures in the model. It uses general information also entered in English, and states in

English any assumptions and facts used to find the solution to the problem given. If the solution is found, it is also communicated in English.

STUDENT could be extended in many ways within the framework developed. For example, much more sophisticated sentence parsing routines could be used. This would enable STUDENT to extract "simple sentences" from much more complicated grammatical structures. In addition, it might help in the identification of a phrase which is a paraphrase of another. This identification is the most difficult task STUDENT attempts, and the methods used are heuristic and ad hoc. Better methods making more use of the meaning of the words in the phrases should be found. Because of the current method of identifying variables, a stored schema or equations can be used only once. This should be improved.

I think we are far from achieving a program which understands all of English. However, within its limited area of competence, STUDENT has demonstrated that it has a good "understanding" of English, and I think that limited programs such as this may actually prove to be useful tools in their own right.

CS-TR Scanning Project
Document Control Form

Date : 11/30/95

Report # AIM-66

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
☐ Laboratory for Computer Science (LCS)

Document Type:

☐ Technical Report (TR) ☒ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 36(40-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☒ Single-sided or
☐ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☒ Other:_____

Check each if included with document:

☐ DOD Form ☐ Funding Agent Form ☐ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP: (1-36) UNTITLED TITLE PAGE	2-36
(37-40) SCANCONTROL, TARGETS (3)	

Scanning Agent Signoff:
Date Received: 11/30/95 Date Scanned: 12/6/95 Date Returned: 12/7/95

Scanning Agent Signature: Michael W. Cook

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 12/6/1995

M.I.T. Libraries
Document Services

View publication stats