Every boy is a person.
It understands.
A finger is part of a hand.
It understands.
Each person has two hands.
It underlines.
How many fingers does John have?
I assume that means has as parts.
John is a boy.
It understands.
How many fingers does John have?
I assume that means has as parts.
Each hand has 5 fingers.
It underlines.
How many fingers does John have?
I assume that means has as parts.
Every hand has 5 fingers.
It underlines.
How many automobiles does John have?
I assume that means has as parts.
John is a person of the United States.
It underlines.
The boy is just to the left of the table.
S cognos is a table.
It understands.
The lamp is just to the left of the table.
S cognos is a lamp.
It understands.
The above statement is impossible.
The table is to the right of the chair.
S cognos is a chair.
It understands.
What is the relative position of a person?
The left-to-right order is as follows:
S cognos (any table).

Figure 1. Sample Conversation.

kinship information. Of course, this "tree" model is not a good representation for other kinds of relationships.

Darlington's translation to symbolic logic. This program shows how certain English problems can be translated into logical terms and attacked by formal methods. As with Lindsay's system, the approach used here has limited applicability.

Bennett's computer program for word relations. This system, in which words are related by means of the English formats in which they appear, extracts and stores semantic information in a limited and somewhat awkward way.

7. Bobrow's "STUDENT". This new system deals with linguistic problems which arise while analyzing those English statements which describe mathematical relationships. The program translates "word problems" directly into algebraic expressions.

Each of the above systems has certain interesting features, some of which have influenced the design of SIR. The goal of the present work, however, has been to find and demonstrate a question-answering mechanism which is highly automatic, widely applicable, and reasonably efficient; one which overcomes many of the limitations of the above systems. An appropriate model is the key to this mechanism.

III. THE SIR MODEL

The SIR model is the collection of data which SIR subprograms may refer to in the course of question-answering. It is a "semantic" model in two senses: 1) The information stored in the model is intended to approximate the linguistic "semantic content" or "meaning" of English text; 2) The structure of the model is derived from the "semantic" model structures of mathematical logic.

Many linguists and logicians have considered problems of "semantics" and offered definitions of "meaning". Few of these discussions are sufficiently concrete to be suitable for computer implementation. However, the idea of representing meaning by word associations, suggested by several previous authors, has been adapted as the basis of the SIR model. The resulting word-association structure of the model is general enough to be useful in a wide variety of subject areas; yet, the stored information is specific enough to provide convenient accessibility of relevant facts and, therefore, efficient question-answering.

STRUCTURE OF THE MODEL

The SIR model is structured by means of property-lists (sometimes called description-lists). A property-list is a sequence of pairs of elements, and the entire list is associated with a particular object. The first element of each pair is an attribute applicable to a class of ob-
jects, and the second element of the pair is the value of that attribute for the object described.

If an English statement asserts that relation \( R \) holds between objects or classes named \( x \) and \( y \), or equivalently, that the word "\( x \)" is associated with the word "\( y \)" in a manner specified by the word "\( R \)", then this relationship is represented in the SIR model by attribute-value pairs placed on the property-lists of both \( x \) and \( y \). Each attribute specifies a relation, and the value paired with the attribute (the value of the attribute) indicates which other objects are related to the described object by means of the specified relation, i.e., which other words are associated with the described word in the way specified by the attribute word.

Since, in general, relations are not symmetric, relation \( R \) must be factored into two relations \( R_1 \) and \( R_2 \) so that, if \( R \) holds between \( x \) and \( y \), one can say that \( y \) stands in relation \( R_1 \) to \( x \) and \( x \) stands in the inverse relation \( R_2 \) to \( y \).

For example, consider the sentence, "Every boy is a person." SIR takes this sentence to mean that a set-inclusion relation holds between "BOY" and "PERSON". The factor relations \( R_1 \) and \( R_2 \) are "SUPERSET" and "SUBSET", respectively. The fact specified by the above sentence can be represented in the model by the attribute-value pair "((SUPERSET, PERSON)" on the property-list of "BOY", and the pair "((SUBSET, BOY)" on the property-list of "PERSON".

An attribute can only appear once on a given property-list. However, the value of an attribute may be a list containing several object names. For added flexibility, elements of these value lists may themselves be in property-list format so that they can hold descriptive information as well as object-names. The first item on such sub-property-lists is the flag "PLIST" which indicates that a property-list follows.

For example, after the system learns that "A person has two hands" and "A finger is part of a person", the property-list of "PERSON" would contain the attribute-value pair:

\((\text{SUBPART}, (\text{PLIST} \text{ (NAME, HAND)} \text{ (NUMBER, 2)}) \text{ PLIST} \text{ (NAME, FINGER)}))\) These general links are the principal mechanism for structuring the model.

**IV. NATURAL LANGUAGE INPUT**

The language which is most convenient to the users of SIR is natural English; therefore, the input and response languages of the SIR system should be reasonably close to natural English. Since its internal information representation is a relational model, SIR is faced with the difficult problem of extracting word associations from natural language.

The work reported in this paper is concerned with the ability of a computer to utilize relational information in order to produce intelligent behavior. The linguistic problem of transforming natural language into a usable predicate form is only of peripheral interest here. The following outlines a superficial but adequate method for solving this linguistic problem in the present context. Reference 14 contains further details of this method.

SIR recognizes only a small number of sentence forms, each of which corresponds to a particular relation. The input language is defined by a list of formats. Each format is a string of constants and variables, and an applicability test is associated with each variable. An input sentence is recognized if it is matched by the constants in some format, and if its substrings corresponding to each variable in that format satisfy the corresponding tests. Special functions associated with each format then extract appropriate word associations from recognized sentences, and perform the desired storage or retrieval operations in the model.

For example, the sentence, "Every boy is a person", is recognized by the format, "\( x \) is \( y \)". Applicability tests check that the substrings corresponding to \( x \) and \( y \) are each two words long, the first of which is a member of the set \{ a, an, every, any, each \}. The associated function then creates set-inclusion links between "BOY" and "PERSON" in the model.

Some formats do not uniquely determine word relations. As an example of how such ambiguity may be treated, SIR considers the verb "to have" as meaning either "to have attached as parts" or "to own", e.g., "John has ten fingers" vs. "John has three automobiles". The function associated with the format "\( x \) has \( n \) \( y \)" must resolve this semantic ambiguity before it can operate on the model. The ambiguity
is resolved, as described in Section V, on the basis of word-associations in the model which were created because of previous, unambiguous input sentences.

SIR always makes reports of its actions. The conversation of Figure 1 was produced by an abbreviated-response mode in which only directly relevant responses were printed. Alternatively, the system can provide a running commentary of its activities. Although less readable, this full-response mode was a significant debugging aid. Figure 2 shows the dialogue of Figure 1 in that mode.

V. THE SIR PROGRAM

SIR is programmed in the LISP language, a list-processing computer language well suited for model building and searching procedures. The operation of the program is described in detail in. Here we we shall observe

Figure 2. Sample Conversation in Full-Response Mode.

From the collection of the Computer History Museum (www.computerhistory.org)
the system's behavior by means of annotated examples.

Each part of Figure 3 is a conversation between a person and SIR designed to illustrate SIR's ability to "understand" a different relation or group of relations. Each part starts with a "clean" memory, i.e., an empty model and no knowledge of vocabulary except for format constants. The following notes, keyed to Figure 3, should help clarify some of the responses:

a1. The response, "I UNDERSTAND", indicates that some desired link has been created or discovered in the model.
a2. "IS" and "IS AN EXAMPLE OF" are equivalent formats in this context.
a3. In general, question-answering routines in SIR can perform all necessary logical deductions. "Q" should be read as a question mark.
a4. The program responds "SOMETIMES" to the question, "Is an \( x \) a \( y \)?" if it can deduce, from existing linkages, that \( y \) is a subclass of \( x \).
a5. "Insufficient Information" is the most common report of failure. The present system does not handle deductions from negative premises, e.g., "Every boy is not a girl."
b1. Absence of an indefinite article indicates that MAX is an individual and, therefore, is an element, rather than subclass, of IBM-7094.
b2. "THE BOY" requires the existence of a unique element of the class "BOY". SIR assigns "G" names to anonymous individuals.
b3. "THE BOY" is assumed to be G02840.
b4. JOHN and G02840 may be different boys, making "THE BOY" ambiguous.
c1. Two words are linked by the attribute "EQUIV" if they are different names for the same object. Deduction procedures must allow for possible equivalences.
d1. In these cases attributes on the property-list of the name of a set may describe properties of every element of the set, rather than of the set as a whole. The hyphens in "PAIR-OF-RED-SUSPENDERS" are necessary to avoid confusing the format recognition scheme.
d2. Each question-answering program may check for certain special cases before attempting standard deduction procedures.
e1. "Specific" information appears on the property-lists of individuals rather than of sets. Deduction procedures must use both general and specific information.
f1. See d1.
f2. See d2.
f3. The system discovers that a NOSTRIL is part of a PERSON, and then answers the question, "Is a living-creature a person?"
h1. In this and future sentences, the system reports that the sentence form is ambiguous (because "have" may mean either "have as parts" or "own"), but it has been able to make a "reasonable" assumption about the intended meaning and is proceeding from there. (See description of Figure 4b.) In this case, "How many" cannot be answered unless a part-whole relationship can be established first.
h2. "NUMBER" is an attribute which is placed on sub-property-lists associated with part-whole relationships.
h3. Here a "NUMBER" attribute is missing along the chain of links which established the part-whole relationship.
i1. "JUST ... " requires adjacency.
i2. In this section new links are created only if they are consistent with already "known" facts.
i3. A "WHERE" question requests location information obtainable from direct links only.
i4. "WHAT IS THE POSITION" requests the construction of a linear ordering of objects, as far as available linkage information permits. Inner parentheses in the order list indicate adjacent objects.

SPECIAL FEATURES

Figure 4 illustrates three special features of the SIR system.

a) Exception principle: General information about "all the elements" of a set is con-
Figure 3. Selected Conversations.
A COMPUTER PROGRAM WHICH “UNDERSTANDS” 583

A. EXCEPTION PRINCIPLE

A. EXCEPTION PRINCIPLE

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Understanding</th>
</tr>
</thead>
<tbody>
<tr>
<td>I... THERE ARE 5 FINGERS ON EVERY HAND</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... THERE ARE TWO HANDS ON A PERSON</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... A BOY IS A PERSON</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... TOM IS A BOY</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... DICK IS A BOY</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... HARRY IS A BOY</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... TOM HAS NINE FINGERS</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... DICK HAS ONE HAND</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... HOW MANY FINGERS DOES DICK HAVE</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... HOW MANY FINGERS DOES HARRY HAVE</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... HOW MANY FINGERS DOES JOE HAVE</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... HOW MANY FINGERS DOES TOM HAVE</td>
<td>I UNDERSTAND</td>
</tr>
</tbody>
</table>

B. RESOLVING AMBIGUITIES

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Understanding</th>
</tr>
</thead>
<tbody>
<tr>
<td>I... JOHN IS A PERSON</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... DICK IS A PERSON</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... A CHAIN IS PART OF A BICYCLE</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... THE POIN-GEY HAS A CHAIN</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... JOHN OWNS A CHAIN</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... DICK HAS A CHAIN</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... THE ABOVE SENTENCE IS AMBIGUOUS</td>
<td>I UNDERSTAND</td>
</tr>
</tbody>
</table>

C. STREAMLINING LINKAGES

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Understanding</th>
</tr>
</thead>
<tbody>
<tr>
<td>I... JOHN IS A PERSON</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... JOHN IS A TECH-MAN</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... JOHN IS A BOY</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... JOHN IS A STUDENT</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... JOHN IS A BRIGHT-PERSON</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... EVERY BOY IS A PERSON</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... EVERY TECH-MAN IS A PERSON</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... EVERY TECH-MAN IS A BRIGHT-PERSON</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... EVERY BRIGHT-PERSON IS A PERSON</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... EVERY STUDENT IS A BRIGHT-PERSON</td>
<td>I UNDERSTAND</td>
</tr>
<tr>
<td>I... EVERY STUDENT IS A PERSON</td>
<td>I UNDERSTAND</td>
</tr>
</tbody>
</table>

END OF EVALQUOTE, VALUE IS NIL

FUNCTION EVALQUOTE HAS BEEN ENTERED, ARGUMENTS

STREAMLINE (JOHN)

END OF EVALQUOTE, VALUE IS NIL

END OF EVALQUOTE, VALUE IS NIL

Figure 4. Special Features.
sidered to apply to particular elements only in the absence of more specific information about those elements. Thus, it is not necessarily contradictory to learn that “Mammals are land animals”; and, yet, “A whale is a mammal which always lives in water.” In the program, this idea is implemented by always referring for desired information to the property-list of the individual concerned before looking at the descriptions of sets to which the individual belongs.

True “understanding” is frequently characterized as an ability to reason appropriately with facts which appear contradictory or paradoxical—an ability generally considered to be beyond formal logical procedures. Here, on the other hand, we see that a simple algorithm used in conjunction with the SIR model is sufficient to produce reasonable, human-like conversational behavior in just such a paradoxical situation.

b) Resolving ambiguities: The criteria used by the program to decide whether “has” in the format “x has y” should be interpreted “has as parts” or “owns” are the following:

Let P be the proposition, “either y is known to be part of something, or y is an element of some set whose elements are known to be parts of something.”

Let N be the proposition, “either y is known to be owned by something, or y is an element of some set whose elements are known to be owned by something.”

1) If P \& \sim N, assume “has” means “has as parts.”
2) If \sim P \& N, assume “has” means “owns.”
3) If \sim P \& \sim N, give up and ask for re-phrasing.

Let P’ be the proposition,

\[ (u) [(y \text{ known to be part of } u) \lor (y \text{ an element of some set whose elements are known to be parts of the elements of } u)] \land (w)[(u \in w \lor u \subseteq w) \land (x \in w \lor x \subseteq w)]. \]

Let N’ be the proposition,

\[ (u) [(y \text{ known to be owned by } u) \lor (y \text{ an element of some set whose elements are known to be owned by the elements of } u)] \land (w)[(u \in w \lor u \subseteq w) \land (x \in w \lor x \subseteq w)]. \]

4) If P’ \land \sim N’, assume “has” means “has as parts.”
5) If \sim P’ \land \sim N’, assume “has” means “owns.”
6) Otherwise, give up and ask for re-phrasing.

These criteria are simple, yet they are sufficient to enable the program to make quite reasonable decisions about the intended purpose in various sentences of the ambiguous word “has”. Of course, the program can be fooled into making mistakes; e.g., in case the sentence, “Dick has a chain”, had been presented before the sentence “John owns a chain”, in the above dialogue. However, a human being exposed to a new word in a similar situation would make a similar error. The point here is that it is feasible to automatically resolved ambiguities in sentence meaning by referring to the descriptions of the words in the sentence—descriptions which can automatically be created through proper prior exposure to unambiguous sentences.

c) Streamlining linkages: All question-answering (model-searching) functions which involve references to set-inclusion or set-membership relations must “know” about the basic properties of those relations; i.e., those functions must have built into them the ability to apply theorems like

\[ x \subseteq y \land y \subseteq z = x \subseteq z \]

\[ a \in x \land x \subseteq y = a \subseteq y; \]

otherwise, the functions would not be able to make full use of the usually limited information available in the form of explicit links. On the other hand, since the functions involved will be “aware” of these theorems, then the set of questions which can be answered is independent of the presence or absence of explicit links.
A COMPUTER PROGRAM WHICH "UNDERSTAINS" 585

which provide the information to the right of the "=" provided the information to the left of the "=" is available. The "STREAMLINE" operation starts with the object x which is its argument, and considers all objects linked to x, directly or indirectly, through set-inclusion or set-membership. All explicit links among these objects which can also be deduced by use of the above-known theorems are deleted. A response of the form "(I FORGET THE SET-INCLUSION RELATION BETWEEN y and z)" indicates that whatever links were created by some sentence of a form similar to "(EVERY z IS A y)" are being deleted, and the space they occupied is being made available for other use.

VI. EXTENSIONS OF SIR

Four major obstacles prevent the immediate expansion of a SIR-like system into a large, practical question-answering system: input language, memory size, search time, and subprogram interaction.

SIR's input language consists of sentences which match a number of simple formats. For convenient use, a more general system should accept a larger subset of natural English. Various research groups are studying this difficult problem of translating from English into a formal language. The research effort described here concentrates, instead, on the representation and retrieval problems which will remain after the input language has been formalized.

The immediate reason for terminating the current project was the lack of computer memory for additional programs and data. However, at that point problems involving search time and subprogram interaction had also become significant. Progress in solving these latter problems is necessary so that we may be ready to use profitably the larger, better organized memories which will undoubtedly become available before long.

In SIR, the time required to search for the existence of particular relational links in the model increases rapidly with both the number of individual elements which can be linked and the number of different relations which can do the linking. Of course, this exponential growth of tree structured search space is a familiar phenomenon in theorem proving, game playing, and other problem solving areas. General tree searching heuristics such as partitioning a tree into non-intersecting branches, searching for a path simultaneously from two endpoints toward an unknown common point, and generating intermediate "stepping stones" to use in a long search, must be further developed for application to the SIR model.

The most basic obstacle to enlarging SIR is the problem of subprogram interaction. The present system contains a separate subprogram for performing each different information retrieval task. This diffuse program structure was convenient during the early development of the system because it facilitated modifying component programs to experiment with different model structures and different search and deduction schemes. However, each new relation added to such a system may affect the question answering procedures associated with those relations already included. Since these procedures are buried in various subprograms, the addition of the subprograms associated with the new relation frequently necessitates program changes throughout the system. Because of these "interactions," the resulting program becomes more awkward and more difficult to generalize as its size increases. Future versions of semantic question answering systems must avoid this increasingly complex program organization. One solution might be based on a proposed "formalized" question answerer, outlined below and described in more detail in reference 14, which increases the question answering power as well as simplifies the program structure of the system.

A FORMALIZED QUESTION ANSWERER

The formalized question answerer consists of the following components:

1. a formal system whose sentences correspond in a well defined way to "yes or no" questions; and a theorem-proving program which can determine whether well-formed sentences are "true," i.e., whether the corresponding questions should be
answered "yes," on the basis of information in the model.

2. A model similar to the SIR model but containing, in addition to links relating objects, axioms and deduction rules of the formal system for the use of the theorem proving program.

3. A programming language for specifying question-answering procedures which are more complex than truth-testing.

The formal system has the same structure as the first order predicate calculus except that the domain of all quantifiers is the set of objects described in the model. Since this set is finite, the system is logically equivalent to the propositional calculus. The basic predicates of the formal system can be initially just those which are needed, along with quantifiers and sentential connectives, to express any question which the SIR system is capable of answering. It can be shown that only a few basic predicates are needed to enable the formal system to express a great many questions. Also many interactions between predicates, which created programming difficulties in SIR, are automatically represented by the structure of the sentences in the formal system.

The formal system is decidable; therefore a program could be written which would answer any well-defined question on the basis of axioms and facts in the model. However, since the model might be large, such a program might be quite slow. On the other hand, a heuristic program could be written which would attempt to answer questions only on the basis of the most relevant data, where "most relevant" is defined in terms of the structure of the model. Such a program could conceivably be as efficient as the special purpose question answering subprograms of SIR.

The model in the formalized question answerer is the same as the SIR model except for containing an additional class of data. The described objects in this model can be names of real objects or classes of objects, or names of basic predicates in the formal system. The property list of a predicate contains all axioms or special deductive procedures associated with that predicate. The theorem proving program would have to perform in accordance with this data in the model. Thus the user could "tell" the system how to change its question answering procedures, whenever such changes were desired, simply by changing the content of the model.

A theorem proving program can only answer "yes or no" questions. However, some of the questions which SIR can answer require the system to perform more elaborate information retrieval tasks. The power of a general purpose symbol manipulation computer language such as LISP must be available for specifying these computational procedures since new question types may require, in the answering process, unanticipated kinds of data manipulation. However, these procedures should be made as easy to write and to understand as possible. In particular, the full power of the theorem proving program should be available within the procedure specification language.

For example, suppose the theorem prover is represented by the LISP function "theorem [x]," whose value is TRUE if x is a sentence in the formal system corresponding to a question whose answer is "yes," and FALSE otherwise. Then suppose that in the course of the procedure for answering the question, "what is the relative position of x?" it is determined that y is to the right of x and also that a z is to the right of x. The procedure could then contain the statement,

if theorem [(a) [axz \right[ a \cdot x ] \right[ y \cdot x ]]] then go[A] else go[B] where A and B are appropriate further instructions in the procedure. The procedure writer need not consider how to answer the question, "is a z between x and y?" for the theorem function, i.e., the theorem proving program, will do that for him.

Space does not permit a more detailed description of the proposed formalized question answerer. However, it should now be clear that such a system has all the question answering ability of SIR and accepts a much larger class of questions. More important, new relations can be added to the formalized system and the axioms of its proof procedure can be modified without any significant reprogramming, thus overcoming the "subprogram interaction" obstacle to the expansion of SIR.
VII. CONCLUSIONS

THE MODEL

The power of SIR's question answering subprograms is due primarily to the flexible property-list structure of the model.

SIR is not unique in permitting facts to be automatically added to or excised from the system. Several existing computer systems, e.g., airline reservation systems, permit dynamic fact storage and retrieval. However, the existing systems generally depend upon the use of a fixed, unique representation for the information involved. A response generally is determined by the explicit presence of absence of a particular item of data.

In SIR, on the other hand, although the model is general enough to contain a wide class of data, it is organized so that subprograms may search it for any facts from which an appropriate answer may be deduced. Because of the way the deduction subprograms use the model, a fact may be represented in many different equally effective ways. E.g., the system "knows" that the statement, "a finger is part of John" is true if (a) there is an explicit part-whole link from FINGER to JOHN; or if (b) there are links by means of which the retrieval programs can deduce that a finger is part of a person, and John is a person; or if (c) there are links by means of which the retrieval programs can deduce that a finger is part of a hand, and a hand is part of John; etc. Thus the use of this model facilitates question answering even in the absence of complete, explicit data. In addition, the system can automatically translate from one representation to another having some advantages. E.g., the "streamline" operation described in Section V reduces storage space requirements by removing redundancy in the representation, without necessitating any changes in the operation of the system.

VALUE OF PROGRAMMING

Many of the results and conclusions written after the development of a large computer program such as SIR frequently appear as if they could have been established without the tedious effort of programming. This is rarely true, and in fact, new systems which are described as complete "except for the programming" usually require fundamental modifications if and when they are translated into operating programs. The reasons for the importance of actually writing the program include the following:

(a) Without a program it is difficult to tell whether the specifications for a system are really complete and consistent. The process of building an operating system makes one aware of major problems which might otherwise remain unnoticed.

(b) Programming not only turns up fallacies in the specifications for a system, but also usually suggests ways for avoiding them and improving the system. A completed "debugged" programmed system usually turns out to be a compromise between the system as it was originally specified, a simpler system which was more feasible to actually construct, and a more elaborate system whose new features were thought of during programming. This resulting system is frequently as useful as and certainly more reliable than the originally specified system, and in addition it may suggest the design of even more advanced systems. With SIR, for example, methods for implementing the "exception principle" and the resolution of ambiguities arose from the design of the basic question answerer, and the specifications for the formalized system of Section VI were based largely on properties of the final, working SIR system.

(c) The programming process frequently turns up insights which might not otherwise be discovered.

(d) Finally, the resulting program provides at the same time a demonstration of the feasibility of the ideas upon which it is based, a measure of the practicality of the system in terms of time and space requirements, and an experimental device for testing variations in the original specifications.

NEXT STEPS

The present SIR system, and its formalized version discussed in Section VI, are proposed as first steps toward a true "understanding" machine. Further steps will involve developing
better means by which a computer system can add to its store of “knowledge” and can “intelligently” select relevant data from that store to use in particular problem solving tasks. These goals are embodied in the “advice taker” problem, which is that of designing a machine whose operation is controlled by “advising” it, in a suitable English-like language, of desired procedures or results.

One type of “advice taker” is a program which can do any of a particular class of problems, such as writing other computer programs, in accordance with simple instructions. Simon is working on such a program-writing program which accepts a broad range of descriptive English sentences as its input.

SIR represents a different approach. Instead of developing various special purpose advice takers, we attempt to build a single, general program which can do any task provided that program is properly controlled by information in its model. “Giving advice” then requires only the process of inserting control information into the model. In a sense, this program is simply an interpreter of information provided in the easily changeable model.

The SIR model provides its programs with information about the truth of particular relations between specific objects. The model in the formalized system of Section VI also provides the “theorem proving program” with axioms which describe properties of relations and interactions between relations. A next generalization would involve adding to the model information which specifies and controls theorem proving and model searching procedures for the program.

Ultimately the “intelligent” machine will have to be able to abstract from the information in its model, “realize” the necessity for additional action, and create the necessary instructions for itself. The design of such an “artificial intelligence” awaits the development of automatic concept formation and inductive inference systems as well as the generalizations of SIR described above.

REFERENCES


A QUESTION-ANSWERING SYSTEM FOR HIGH SCHOOL ALGEBRA WORD PROBLEMS*

Daniel G. Bobrow
Massachusetts Institute of Technology, Cambridge, Massachusetts

I. INTRODUCTION

The aim of the research reported here was to discover methods for building computer programs which can understand and communicate with people in a non-trivial subset of English. A computer program understands a subset of English if it accepts input sentences which are members of this subset, and answers questions based on information contained in the input. We describe in this paper a semantic theory of discourse, and utilize a first approximation to the analytical portion of this theory in the STUDENT question-answering system, a program which understands a subset of English in the sense defined above.

The STUDENT system, programmed in METEOR and LISP, accepts as input high school algebra word problems expressed within a restricted but comfortable subset of English. For example, STUDENT will accept the following problem statement:

"The price of a radio is $69.70. If this price is 15 percent less than the marked price, find the marked price."

After some computation STUDENT will respond:

"The marked price is 82 dollars."

If needed, STUDENT has access to a store of "global" information not specific to any one problem, and can retrieve relevant facts and equations from this store of information. For example, when solving the problem:

"If 1 span equals 9 inches, and 1 fathom equals 6 feet, how many spans equals 1 fathom?"

STUDENT retrieves and uses the fact that 1 foot equals 12 inches, and prints the answer:

"1 fathom is 8 spans."

STUDENT is embedded in the M.I.T. Project MAC time-sharing system. Therefore, as a last resort, when it can not solve a problem, STUDENT requests and can obtain immediate help from the questioner.

A number of other English language question-answering systems have been constructed; the most closely related work was that of Green, Lindsay, and Raphael. A critical analysis of this related work and criteria for evaluating a question-answering system may be found in the author's thesis. Simmons gives a descriptive survey of systems which answer English questions.

* The work reported here was supported in part by the M.I.T. Computation Center, in part by the M.I.T. Research Laboratory of Electronics, and in part by Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under ONR Contract Number Non-