WORKING PAPER
ALFRED P. SLOAN SCHOOL OF MANAGEMENT

CONTROL STRUCTURE FOR A QUESTION-ANSERING SYSTEM

Ashok Malhotra

January 1974   #693-74

MASSACHUSETTS
INSTITUTE OF TECHNOLOGY
50 MEMORIAL DRIVE
CAMBRIDGE, MASSACHUSETTS 02139

CONTROL STRUCTURE FOR A QUESTION-ANSERING SYSTEM

Ashok Malhotra

January 1974   #693-74

HD28
,M414
no.693-74

INTRODUCTION

Despite my best efforts at explanation, my grandmother still thinks of the computer as a machine that answers questions. To her, this is a manifestation of intelligence far more powerful than doing "sums"; for while everybody answers questions, few can do "sums"! It is not surprising to me, therefore, that question-answering systems have as long a history in artificial intelligence as problem solving.

This paper describes the design of a proposed question-answering system that will answer questions from a corporate data base. We will discuss the rationale behind the proposed system and the design decisions that follow from the initial assumptions. Further, we will discuss the question-answering strategies developed for this system and the basic principles underlying them.

EARLY QUESTION-ANSWERING SYSTEMS

McCarthy (1) seems to have been the first to realize the importance of verbal learning and reasoning. In 1958 he described the design for an "Advice Taker"; a machine that would accept natural language input and be able to solve problems

involving deduction on given premises. This paper called attention to a number of basic issues and it is surprising how much of contemporary A.I. work can trace its roots to this seminal contribution. One of the problems it discussed was natural language understanding and Phillips (2), inspired by Chomsky's work, built a system that answered questions by matching the parse of the input with those of the declarative sentences it had received earlier as data. BASEBALL (3), built with Chomsky's collaboration, was an ingenious system that provided baseball statistics in response to questions in a limited subset of English. It operated on a data base in a prestructured tree format and was thus able to separate the problems of creating a data base from those of answering questions about it.

SYNTHEX (4) was an ambitious system that contained the text of a childrens encyclopedia as a data base. An index was prepared listing the location of each "content word". Sentences that could be relevant to the question were then retrieved using this index and subjected to an analysis procedure to screen out the irrelevant sentences.

The analysis procedure was based on simple syntactic considerations, however, and was not very effective. Further, the system was incapable of deduction and could not, therefore, consolidate the information contained in a number of sentences. Since the grammar was independent of the program it could easily

be extended but the program suffered from a lack of semantic knowledge and the underpinning of a model of the world.

SAD-SAM (5) and SIR (6) accepted information in the form of a restricted set of sentence types and created a semantic network of objects and relations. This network was then used to answer questions. SAD-SAM was resticted to familial relations while SIR was capable of accepting and analysing set, part, position and ownership information. It was relatively simple to increase the input formats in SIR but the semantic analysis routines were considerably more difficult to extend.

Each of these early systems attacked more than one of the following problems.

  a. Creation and modification of a world model from
      (restricted) natural language input.

  b. Answering questions from a data base.

  c. Solution of simple problems using deduction on data base
      elements.

In addition, most of these systems took on the task of understanding some subset of English! We now realize how difficult each of these problems is and can admire the ingenuity of these early systems that achieved encouraging results by more or less frontal attacks.

LATER SYSTEMS

It is not surprising therefore, that later systems were more limited in their objectives and tended to focus on only one of the above problems. We shall describe only the natural language question-answering systems here although there has been a great deal of development in the other areas as well.

W. A. Woods developed a "transition network" grammar for natural languages and implemented a parser to analyze sentences in accordance with it. He used this parser in a system to answer questions from the data base of an airline schedule book (7) and later, from the chemical analysis of the moon-rocks (8). Woods' mainly syntactic parser is very powerful, although its efficacy is limited by its rudimentary semantic knowledge. His data bases are real, if simple, and he has been able to demonstrate the practicability of computer-based question-answering systems in real world situations. "The prototype [of the moon-rocks system] was run twice a day for three days ... and during this time the lunar geologists .. . were invited to ask questions of the system. ... During this demonstration 80% of the questions which were asked and that fell within the scope of the data base were parsed and interpreted correctly in exactly the form in which they were asked ..."

A question and command system implemented by Winograd (9) for the world of a set of children's blocks on a table-top demonstrates powerful capabilities of sentence analysis and

comprehension. The simplicity of the world contributes in no small part to the impressiveness of the system, however. Further, most of the knowledge in the system is encoded as procedure and this makes it difficult to extend.

In developing a novel approach to computer-aided instruction Carbonnel developed the prototype of a knowledge-based system that was capable of answering questions in addition to asking and evaluating them. A limited system based around knowledge of the geography of South America is operational (10) with a limited subset of English. The system is undergoing further development.

A QUESTION-ANSWERING SYSTEM FOR MANAGEMENT

The negligible impact of computer-based systems on other than routine decision-making is too well known to need documentation. The reasons behind this are also fairly well agreed upon: computers are difficult to communicate with and must be instructed in excrutiating detail. Although powerful, they operate at a level of specificity and with a series of elaborate conventions that the manager cannot tolerate.

To be useful, a computer system must "understand" a great deal about the environment within which the manager operates. Furthermore, it should not require procedural programming and must be capable of performing functions that are useful to the manager in his day-to-day work. Starting with these premises we proposed a question-answering system as a management aid. We decided to allow questions that required retrieval from a data base as well as the evaluation of functions of retrieved data. A restricted class of pre-programmed models could be executed in the form of subroutines but apart from these, no deduction was permitted.

Starting from these guidelines and the general premise that the system should be easy to learn and use for managers with no computer experience we developed a more detailed set of system specifications, many of which proved to be quite different from earlier question-answering systems.

1.Since it is important that neophyte users should be able to

learn to use the system easily and painlessly the system
should answer questions about itself. (We believe this to be
unique in question-answering systems.)

2. The user should be protected from system errors and should not
   be expected to take any remedial action.

3. Attempts should be made to continue the dialog with the user
   as far as possible. If a question cannot be answered the
   system should attempt to say something relevant and
   intelligent to convince the user that it has understood his
   request. If a question is ambiguous or unclear the system
   should ask a question or make suitable assumptions (which
   should be intimated to the user) and generate an answer.

4. In a real situation it is not possible to answer
   questions from a data base containing only objects and events.
   Aggregate data must be included. For example, sales
   figures cannot be computed by checking and aggregating values
   from the set of individual sales events. The inclusion of
   aggregates, however, gives rise to name ambiguities. This
   problem and its solution are described in a later section.
   (To our knowledge, no existing question answering system
   incorporates aggregate data and the knowledge necessary to
   answer questions about it.)

5. The subset of natural language that can be used to address the
   system should possess adequate power and flexibility.
   Similarly, response times to typical questions should be

reasonable. These criteria have not been formalized because it is impossible to postulate satisfactory conditions a priori. We propose to establish satisfaction along these dimensions experimentally.

We are convinced that a question-answering system would need to have a great deal of knowledge about the business world, the particular setting it was to be used in and about the data base. Minsky (11) makes an excellent case for the need for such knowledge. This too can be considered to be an initial design decision for the system.

Figure 1. is a schematic diagram of the proposed system. Functionally, the system can be divided into two parts, the parser and the processor. These two operating sub-systems rely upon three knowledge bases: a model of the world, a model of the scenario situation and a data base

This paper describes the design of the processor. The parser has been designed and is being implemented by Prof. W. A. Martin (12,13) and the Automatic Programming group at Project MAC. It has a strongly semantic, "case oriented" approach to the analysis of English sentences. (See Fillmore (14) for the theory of case grammar and Celce-murcia (15) for an early implementation.) Similarly, the processor is also knowledge-based and this is its point of departure from earlier question-answering systems, such as Woods' (7,8), and its main strength.
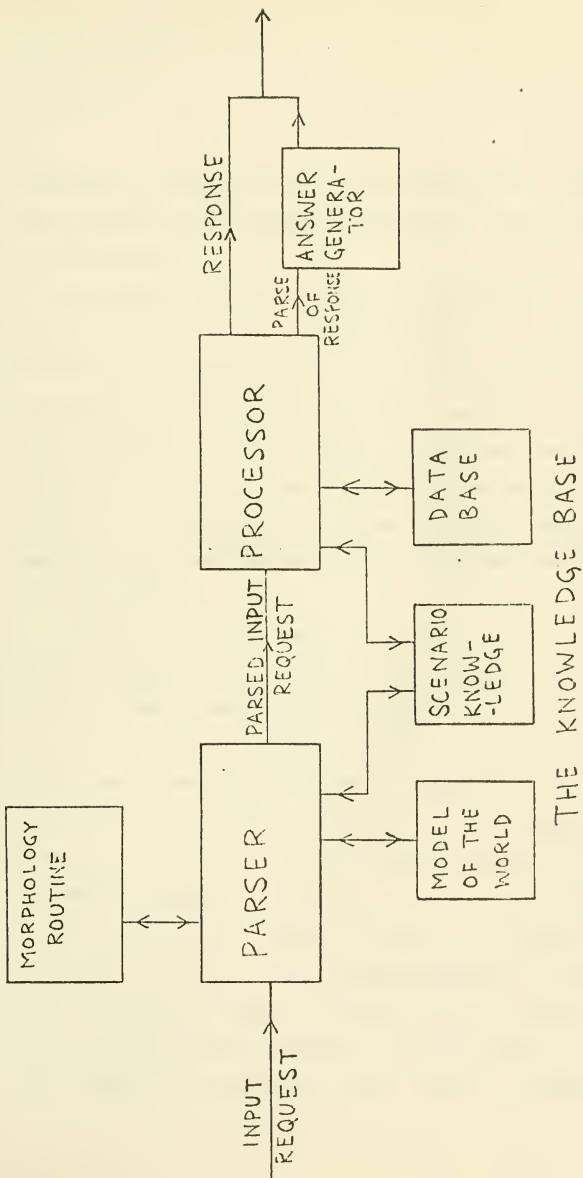
FIGURE 1. SCHEMATIC DIAGRAM OF PROPOSED QUESTION ANSWERING SYSTEM

In addition, the system will operate on a data base considerably more powerful than anything attempted with a question-answering system to date.

The parser examines the input to the system and creates a parse for it. A parse is, in some sense, the meaning of the sentence, but more accurately, it is a set of relations between semantically identified parts of the sentence encoded in a standard format. The processor uses the parse as input and attempts to fulfil the input request and generate an appropriate response. Understanding the sentence therefore, takes place in a general sense within the parser and in a much more specific sense in the processor.

We shall say only a few words about the parser since it is the processor that is central to this paper. The morphology routine examines each word in the input request and checks if it is known to the world model. Unknown words are analyzed to determine whether they are variants of known words or idioms.

If a word cannot be recognized by the morphology routine a message is printed out indicating the offending word and the user is asked to retype his request.

Once the complete sentence is accepted, the parser attempts to find the main verb and to arrange the noun phrases in the sentence as cases of the main verb. Initial prepositions that mark some of the noun phrases assist in this process, but use is also made of the case-frame of the verb which determines the

cases it can take and the meanings of the nouns which determine the cases they can participate in. Most of this knowledge is contained in the world model. Specialized words and word meanings and some idioms are contained in the scenario world.

If the sentence is ungrammatical or cannot be parsed, the parser prints out a message to the user and asks him to rephrase his request. In some cases the parser isolates more than one valid parse. All these parses are transmitted to the processor which uses specialized scenario knowledge to eliminate some of them. If the set of parses cannot be reduced to one, the processor prints out a message stating that the sentence is ambiguous.

THE DATA BASE

This sub-section describes a language for encoding knowledge. Later sub-sections discuss the impact of this organization on question-answering strategies and consider some alternative organizations.

The knowledge representation language (which may be the same as the programming language) must allow the representation of objects and their properties. It must also be able to keep property information of similar objects distinct. In addition, it is convenient if the language has facilities to query the data base and extract information from it. We use a language called MAPL (16) in which a manufacturing corporation called The-

Battery-Company can be represented somewhat as follows:

```
((THE-BATTERY-COMPANY)
  (IS (THE-BATTERY-COMPANY) (KIND CORPORATION))
  (MANUFACTURE (THE-BATTERY-COMPANY) (KIND PRODUCT))
  (SELL (THE-BATTERY-COMPANY) (KIND PRODUCT))
  (EMPLOY (THE-BATTERY-COMPANY) EMPLOYEE))
  (BUY (THE-BATTERY-COMPANY) MATERIAL))
```

This states that THE-BATTERY-COMPANY is a kind of corporation. It buys material, employs employees and manufactures and sells a kind of product. The properties of the product can be inserted into this definition or described separately as:

```
((KIND PRODUCT)
  (IS (KIND PRODUCT) LEAD-BATTERY)
  (COUNT (KIND PRODUCT) 5))
```

This states that the product is five types of lead-batteries (lead-battery being used as a name). Similarly, the corporation could be described further as a sub-chapter 15 corporation in Massachusetts law but this may not be neccessary and (KIND CORPORATION) may be sufficient.

If the product definition is imbedded in the main definition, MAPL will keep it distinct from other product definitions that may exist in the system.

In the same way, we can use MAPL to represent events. The statement "John met Mary at the corner." can be represented as:

```
((MET)
 (AGENT (MET) JOHN)
 (AGENT (MET) MARY)
 (LOCATION (MET) (KIND CORNER))
 (DETERMINER (KIND CORNER) DEFINITE))
```

Mary can be represented as either another agent (as above) or as a kind of object depending on our interpretation of the verb "met".

(TEXT)
(AGENT INST) LONG
(AGENT INST) MANY
(LOCATION INST) KIND CURRENT
(DETERMINER KIND COMMON UNRELATED)

Mary can be represented as a time constant about the adverb
or as a kind of object depending on one interpretation of the
verb "has".

QUESTION-ANSWERING STRATEGIES

A semantic analysis of English questions is presented in the Appendix. This differs from traditional linguistic analysis of questions in that it concentrates on what is questioned and about what rather than upon the syntactic form and the transformations needed to arrive at them. The meaning of the question is, however, determined in part by the syntactic form and therefore syntax is important to the analysis.

The analysis is somewhat selfish in that it analyzes mainly the questions that the proposed question-answering system can expect to encounter. Questions beginning with "why", for example , are not treated, as the system does not expect to be able to answer them.

WH-QUESTIONS

In general, wh-questions (questions starting with "what", "which " etc.) ask for the properties of objects or events. Object properties are questioned by "be" or "have" verbs (possibly modified by tense words) while event properties are generally questioned by the main verb of the event.

After the sentence is analyzed to determine the property questioned the property can be retrieved directly if the object or event exists in the knowledge base and if the property value is explicitly available. At this point we should recognize that the knowledge base may be organized by object or by event. If it

is organized by object then each event will be represented as a property of one of its participants such as its agent. If however, the knowledge base is organized by event objects will be represented as properties of the existence, or "be", verb. A dual organization is also possible in which information is stored both ways. This allows a choice of retrieval paths and world or scenario knowledge can be used to select the path that leads to the least searching. Thus, finding the entity whose properties are questioned can be somewhat complex. After the entity is located the value of the questioned property must be retrieved or deduced. This can be even more complex.

Different question types require variants of the above general strategy but the basic tasks of finding the entity and finding or deducing the value of the questioned property remain more or less central in all cases.

Note that wh-questions requiring objective answers are restricted to the past and present tenses. It may be possible to answer "What was our profit last year?" from the data base but "What will be our profit next year?" requires a predictive model that embodies some level of subjective judgement. Thus, answering wh-questions in the future tense is completely different from answering them in the past or present tense and requires a search for suitable subjective models that may or may not exist.

AGGREGATE DATA

Some properties, such as costs and production figures are best kept as aggregates. The aggregation can be made in several ways and this gives rise to a naming problem. "Sales", by itself, means little; it has to be qualified by the parameters over which it is to be aggregated. These may be manufacturing facility, product, customer, salesman and time-period. Some of these parameters, or keys, as they are sometimes called, have to be picked up from the various cases of the parse, others can be filled in by default and still others may be assigned typical or commonsense values. In fact, there is a powerful, general default that operates in English: IF A PROPERTY IS ATTRIBUTABLE TO SUB-ENTITIES THEN THE ABSENCE OF SUB-ENTITY SPECIFICATION IMPLIES SUMMATION OVER THE SET OF SUB-ENTITIES, I.E. IT IMPLIES A VALUE FOR THE ENTITY AS A WHOLE. In a sales specification, for example, if there is no mention of a product then the sum of sales for all products is required.

Typical values depend on the data in question. Time-periods, for example, can usually be assumed to be the last complete period if unspecified. Knowledge about each data item must contain, therefore, a list of key variables which must be specified before the data can be retrieved. In addition, it must contain information as to where the key variable specifications will be contained in the sentence. This is not too difficult in a case grammer because each kind of information has a special

case assigned to it. For example, time periods occur in the "during" case and customers usually in the "recipient" case. Typical value information must also be contained in the knowledge about each data item but this is better organized by classes of data items rather than for each item.

In the case of "cost" one of the dimensions of aggregation is the reason for or type of expenditure. This gives rise to variously named costs such as "interest expense" and "product transportation cost". As each cost category is stored under a single name in the system it becomes necessary to determine the unique name by operating upon the noun modifiers of the word cost (or expense) and what can be called its context (in "the transportation cost for lead" the context is "lead"). This is done in our system by finding a type of data item and then checking if a subset of it is named by one of the modifiers or the context. This is continued recursively till no further subsetting is possible.

Some sentences can be constructed so that the context serves to determine the unique name as well as provide a key for its retrieval. For example, "Show me the cost for all products" can be assumed to ask for the direct-manufacturing-cost as this is the only cost available that has "product" as a key.

Since different aggregations may be required for different purposes an obvious strategy dictates that data be kept in a fairly disaggregated state and aggregated in response to the

needs of the question. This requires knowledge of how the data is stored and the keys to look for in the sentence to be maintained for each data item. But maintaining data at one level of aggregation may not be sufficient. The aggregation of annual sales for five products from five plants to five customers stored by month requires 1500 probes of the data base! This can mean a long wait at console for the user. Thus, higher levels of aggregation must be maintained as well.

The system should be able to recognize the level of aggregation required to answer the question and use the highest level of aggregation for which data is available. In our system, an "association list" is created every time data has to be retrieved. This list contains the range over which aggregation has to be performed for each key variable on the assumption that the lowest level of aggregation will be used. Before the aggregation is performed, however, the list is processed to check whether data with a higher level of aggregation is available and can be used. If so, the list is modified and the data-name changed to a higher aggregate.

Woods' found that when lunar geologists asked for the analysis of a rock, they wanted, in fact, its analysis for only the ten most common elements. Similarly, a manager may want the distribution of sales by product whenever he asks for "sales". Thus, particular questions may come to have iconic meanings above and beyond thier literal meaning. The present design of the

system does not recognize iconic meanings.

It can be argued that retrieval and aggregation of data are not the direct concerns of the question-answering system and it should restrict itself to extracting the key information from the sentence and passing it on to a data retrieval system that uses knowledge about the structure of the data files to perform the retrieval efficiently. We agree with this decoupling of the problem and hope that it will be used in future implementations.

MODELS

The use of the word model, in management, stems from its meaning as a replica. Typically, models are a set of relationehips that establish the dependencies of target, or result, variables on independent, or decision, variables. The dependent variables are often figures of merit that measure the health of the enterprise or the success of some portion of it and the model is used to assist in the decision-making process by predicting the effects of changes in the independent variables on the dependent variables.

Management models can take a variety of shapes and structures depending on the nature of the process they model and the nature of the decision-making they support. We shall consider only one class of models, those that can be represented by a set of mathematical operations (such as the functions described in the next sub-section) on specified items of data and can, therefore, be encapsulated in a subroutine. (Other types of

models, in particular, those that can be calibrated, specialized and modified by the manager are very important and very useful. They have been excluded here in order to limit the problem to a reasonable size. See, however, Krumland (17).)

Models are usually referred to by name in the question. Input data is not specified unless it is exceptional and neither are the operations to be performed on it. This information must, therefore, be available to the system as a property of the model. Key information relevant to the retrieval of input data must, however, be specified in the question.

A request for model execution will, therefore, cause a search for input names. After these have been located the data is retrieved using the key information and fed into the subroutine that calculates the output value. Although it is always possible to specify models as mathematical functions of data only, this is not always convenient. It is customary, therefore, to specify models that use the output of other models as input. Thus, when the input retrieval routine for models encounters an input that is a model output it calls the model evaluation routine to evaluate it. The structure of correctly specified models ensures that this recursion always terminates.

In addition to an argument list and a subroutine, the knowledge about a model, like knowledge about data items, must also contain information as to where key variable values can be found in the sentence. Each model must also have a definition

that can be used to answer questions such as "How is profit calculated?". In fact, more than one definition is required if questions about various aspects of the model have to be answered properly.

FUNCTIONS

In some cases the property required may be a function of data such as percentage, distribution or average. Unlike models, functions can operate on a variety of data and, therefore, the names of the inputs to the function have to be specified in the sentence along with key specifications for their retrieval.

A number of conventional devices are used to specify the data items on which the function is to operate. For example, if the percentage of a subset of an item is required, the question will name the data item and the subsetting characterictic: "What percentage of plant capacity is utilized?". Similarly, if a distribution is required, the data item will be named along with the key variable or subsetting name. The data will have to be aggregated over all the unspecified variables and retrieved as a function of the specified characteristic.

In general, data names, key specifications and subsetting names occur in different parts of the sentence and the determination of key specifications for each argument must take into account the function to be executed. Consider, "How much did sales increase over 1972?" and "How much did sales increase

in 1972?". To interpret and answer these two questions correctly the system must understand that the arguments to the function "increase" are both of the same type with different key variable values. Typically, only one set of key variables is specified, the others being picked up by default. The defaults, however, depend on knowledge about the function.

Thus, the analysis of questions that ask for functions of data has to depend heavily on knowledge about the function to determine data name and key variable information. Once this has been done, however, the data items can be retrieved and fed into a subroutine that evaluates the function.

Functions are usually specified by name in the question. An exception is "How many" which asks for the count of a set of objects or events. This is, however, an unusual function since it does not operate on data.

DEFINITIONS

Consider the questions "What information do you have about product cost ?" and "What do you know about product cost ?" After product cost is located directly and checked to be a subset of a possession of "you" (information) or located by a search from among the possessions of "you" what does the system reply?

The problem with "know" is similar, although the object known may be called knowledge and the search avoided. In fact, what is required is, in some sense, the definition of product

cost. This is implicit in the question but must be made explicit for the system.

In fact, these examples demonstrate a more general convention in English: IF THE QUESTION IS ABOUT AN OBJECT THEN ITS MOST IMPORTANT CHARACTERISTIC IS REQUIRED. For example, "What were sales in 1972?" asks for the value of sales: value being the principal characteristic of data. "What is a dog?" requires the dominant characteristic of dogs in general, perhaps that they are animals. "What is this dog?", however, asks for the dominant characteristic of this particular dog: perhaps its breed. Carbonnel and Collins (10) represent these "super concepts" for each noun explicitly in the data base.

YES-NO-QUESTIONS

As discussed in the appendix, yes-no-questions, if restricted to the present and the past tense inquire about the truth of stated propositions. Propositions may concern the properties of stated objects and the identity of actors and other particulars about events. Yes-no-questions in the future tense involve predictions and judgements about the future and are, therefore, excluded from the scope of this system.

In a data base organized by nouns, a yes-no-question inquiring about the truth of stated properties of nouns can be answered by selecting the appropriate nouns and matching their properties against the stated properties. We find a recursive

property matching routine very useful for this. It must be remembered that the properties mentioned in the questioned must be checked against those in the data base and not vice versa as the data base may contain properties other than those questioned.

Event matching is somewhat more complex in such a data base. Since events are stored as properties of the agent or the possessor these nouns have to be located in the question and in the data base and their properties searched for the event. Once this is accomplished, or a set of possible events selected, their properties have to be matched against those in the question.

The alternate organization by event, or verb, requires a decision as to whether objects that participate in an event should be stored under the event and/or under the existence ("be") verb. Direct retrieval for identity-questions and yes-no-questions requires both facilities:

"Who robbed the bank?" property of "rob"

"Did Jack rob the bank?" property of Jack

If properties are only stored by event then an indirect retrieval is required for the second question much like the indirect retrieval required for events in a noun-oriented data base.

There seems to be an essential duality between objects and events in the world but depending on the question and the contents of the data base there may be more objects than events or vice versa. Thus, retrieval through one or the other will

lead to less searching. If the length of data files is less important than the amount of searching required in the retrieval process, then a dual organization in which properties are stored under both events and objects and the retrieval path selected, with the help of the knowledge base, to minimize search will yield the best results. In fact, MAPL is designed for the dual organization and automatically stores properties under both the object and the event.

IDENTITY QUESTIONS

Questions that start with "who" or "which" are quite different from other wh-questions and are, in fact, more like yes-no-questions. What is required as an answer to these questions is the identity of the object that satisfies the properties stated in the question and the process of answering can be likened to answering a set of yes-no-questions on a set of candidate objects that are capable of satisfying the conditions in general. "Who" questions ask for the identity of animate objects while "which" questions can ask for either animate or inamimate identities.

Answering routines for identity questions, therefore, start with the selection of a set of candidate objects. The generic name for this set is invariably specified as the main object noun in identity questions with "be" verbs that specify the properties of the required object rather than the event it participated in.

The candidate set is the set of all objects that are "a kind of"
the generic name. "Who questions that ask the identity of the
agent of an event do not give any direct clues to the candidate
set of objects. There is, however, the rule that the objects in
the candidate set must be able to perform the given event and
this can often be used to narrow down the set from the set of all
animate objects. In many data bases the set of animate objects
is fairly small and so a search through all of them is not very
time-consuming.

Once the candidate set is established, the selection process
is much like performing a yes-no-question matching on each event
except that the result of each matching is the identity of the
candidate or "no". The final reply to the question can be
created from the set of identities that have been returned by the
individual yes-no-questions. If this set is the null set then
the appropriate answer is "None of them" for "which" questions
and "No one" for "who" questions.

THE WORLD MODEL PROBLEM

Every data base and every question-answering system embodies
a particular model of the world. Further, it expects questions
about concepts and properties that are sensible in terms of this
model. A severe problem can arise, therefore, if the user has a
model of the world which is at variance with that of the system
in significant ways. For example, our data base contains direct

manufacturing costs and overhead costs for activities that cannot be directly attributed to manufacturing. The overhead costs are not allocated to products, since we feel this is artificial, and therefore break-even points have no meaning in the system. A user who likes to think in terms of break-even quantities will ask for them and may not be able to proceed if the system merely says it can't provide break-even data.

Ideally, the system should be able to realize that there is a discrepancy between world models and since it cannot change its model of the world it should explain it to the user to try to influence his.

The proposed system adopts an extremely simple-minded approach to this problem. It maintains a list of concepts, such as break-even, that it knows belong to variant models of the world. Every time a question asks for one of these concepts it responds by printing out an explanation of its world model and why the question is inappropriate.

IMPERATIVES

Besides asking questions, the user of the system should be able to request services from the system by using commands. This allows the dialog to be much more natural. Commands ask for action and as the actions possible by the system are limited so are the types of commands that may serve as meaningful input to it. The services provided by the system are limited to data

retrieval, model execution and the provision of information about itself. Typical commands to the system, therefore, are as follow:

"Show me the sales to Sears for 1973."

"Display the names of customers with outstandings

of over $5000."

These questions seem to ask for data retrieval.

"Compute the profit for '73."

"Calculate the return on investment last year."

These questions seem to ask for the value of functions of data or the execution of models.

The distinction is specious, however. The structure of the data base will determine what can be retrieved and what has to be computed. The user will, in general, be unaware of the structure of the database and, therefore, his choice of verbs should not be considered significant.

Other verbs have semantic significance and will require special routines to process data and generate answers.

"Compare the distribution of sales to Gulf and Sears by unit."

"Contrast the sales for each quarter of this year and

last year."

"Sort the customers by outstandings."

CONCLUSION

Most of the question-answering systems built to date have

started by devising data base structures and defining question-answering strategies from scratch. To some extent, this was necessary because each system addressed a particular kind of data base and answered limited sets of question types using parsers of very different styles and capability. We have however, learnt a great deal about knowledge representation and question analysis from these systems and it is desirable that future developers of question-answering systems build upon this experience rather than replicate it. We hope that our analysis, limited as it is to our situation and our goals, will be useful to future researchers.

APPENDIX

ANALYSIS OF INTERROGATIVE FORMS IN ENGLISH

This appendix attempts to classify the types of questions and commands possible in English. The classification is not primarily by syntactic structure, since that is the concern of the parser, but rather by semantic structure; what is asked and about what. The structure of the sentence is still rather important, however, as it provides many clues to its semantic function. To the best of my knowledge, such an analysis has not been attempted before. For reasons of brevity presentation is restricted to the sentence types that are expected to be encountered by the question-answering system.

English allows a wide variety of question types that can be analyzed along a number of dimensions. Most basically, questions can be analyzed according to the answers they require: objective, i.e., mathematical and logical functions of known data, and subjective, i.e., requiring the formation of inferences and judgements beyond those possible from the models and data contained in the system. The proposed system will not attempt to answer questions that require subjective answers. This does not mean that it will be unable to answer questions that ask for reasons or opinions, but merely, that it will be restricted to providing such answers that can be constructed by sets of rules and data contained within the system.

Questions requiring objective answers can be further subdivided according to the analysis required to generate the answer and the type

of sentence construction used.

Wh-questions are so called because a word starting in "wh" ("how" is an exception) is used to ask the question. Such questions ask for data or property values. Yes-no-questions, on the other hand, are distinguished by an auxilliary verb, such as "do" or "can", preceding the noun group. These questions enquire about the truth of stated propositions and expect either "yes" or "no" as an answer. Sometimes, of course, the appropriate response is "I don't know".

WH-QUESTIONS

1. Also called adverb questions, these usually start with a question adverb. Each adverb refers to a particular type of attribute. "Why" asks for reason or causality and generally requires subjective answers.

   "Who broke the window ?"   Asking for the identity of the
                                     causal agent.

   "Where did the accident happen ?" Asking for the location.

   "When did the accident happen ?" Asking for the time.

   "What" is special in that the attribute and the object must both be specified for existence ("be" type) verbs.

   "What is the hieght of the Empire State Building ?"

   For other verbs, "what" asks the value of the object. Another case, typically the agent, must also be specified.

   "What did you see ?"   "What did Alex write ?"

Sometimes the object can be assumed as "our" or filled in from
context.
"What were sales in 1972 ?"

"What" can also be used to ask for definitions.
"What is a dog ?"
"What is contribution margin ?"

"How" is somewhat special and comes in four forms:

"How many people attended the ball ?"	Asking for a count.
"How much beer will fill this glass ?"	Asking for quantity.
"How tall is Maria ?"	Allowing the attribute to be named after "How".
"How is profit defined ?"	Asking for method.

In the final form, the exact property questioned depends on the
main verb and, in some cases, even on its object.

Question adverbs can occur by themselves as a sentence provided the
context is known: "Why ?", "Where ?", "How ?".

2.Attribute values can also be questioned by an initial preposition
phrase with a question adverb in place of the noun.

"To whom did you entrust the keys ?"	Asking for recepient.
"From whom did you get the keys ?"	Asking for source.

"With what did John hit him ?"   Asking for instrument.

In some cases the preposition may be shifted to the end.
"What did John hit him with ?"

IDENTITY-QUESTIONS

   Questions starting with "who" or "which" are somewhat different from
other Wh-questions in that they ask for the identity of objects whose
properties satisfy certain criteria.
   "Who broke the window ?"
   "Who is our largest customer ?"
   "Which plants had costs over 10% of budget last year ?" The first of
these three questions requires a selection from among all animate
objects. This is the case with "who" questions with semantic verbs.
The second question requires a selection from the set of customers.
Specification of a common noun that determines the set of objects to
be selected from is typical of "who" questions with "be" verbs.
(questions of the type "Who is Guru Maharaj Ji ?" are an exception.)
"Which" questions always specify a common noun which determines the
set of objects to be searched.

YES-NO-QUESTIONS

   Yes-no-questions come in a variety of forms. If restricted to
   the past and present tense they enquire whether objects have
   stated properties and capabilities and whether stated events

occurred. Another form is a variation on identity questions.
They ask only if the set of objects that satisfy the criteria
is non-empty.

If the future tense is also allowed, Yes-no-questions take
on a considerably more complex character and use modal
constructions to question stated events and courses of action
are plausible, likely, recommended, neccessary, etc. Future
tense yes-no-questions will not be discussed further in
this appendix.

The auxiliary verbs that signal yes-no-questions may be
semantically empty. If not, they specify the following types
of questions:

Modals (could, would, shall)	The potential or expectation to act.
"Have" type verbs (has, had)	Possession or control over.
"Be" type verbs (is, are, were)	Existence.

The following are typical examples of yes-no-questions:

1. Questions asking about the existence of a property.

   "Can monkeys sing ?"

   "Is the car colored ?"

2. Questions asking the truth of statements.

   "Is contribution margin the difference between list price and
standard cost ?"

   "Are monkeys green ?"

3. Questions asking whether attribute values satisfy certain relation.

   "Is the car red ?"

   "is profit this year greater than a million dollars ?"

   "Is profit this year greater than last year ?"

4. Questions asking whether a command can be carried out.

   "Can you stand on your head ?"

   "Can you show me the profit for 1969 ?"

5. Asking whether there is a non-empty set of objects with

   given properties.

   "Is there a house with three windows ?"

   "Are there two cars are identical ?"

   "Did any plants have costs greater than budget last year ?"

   "Are there two cars that are identical ?"

   Each of the above types of yes-no-questions can be formulated

   in a different manner that specifies alternatives to be selected

   from:

   "Do they have children or don't they ?"

   "Can you show me the profit for 1969 or don't you have that data ?"

   "Is his car green, black or red ?

   Clearly, these questions do not require "yes" or "no" as an answer

   but rather the selected alternative(s) or a negative.

COMPOUND AND COMPLEX QUESTIONS

   The above examples are all simple questions, but questions

that occur as complex and compound sentences can be analyzed
similarly. In complex questions, dependent clauses can be
used to specify time, place or other features of the entity
being questioned.

"What was Sears' profit in the year Ajax lumber went bankrupt ?"
"What car, did your brother say, he wanted us to tell Jane to buy ?"
An important class of complex questions are those for which an
"if clause" specifies hypothetical conditions. Usually these are
parameters to a model; unmentioned parameters being picked up by
default.

"If sales went up by 10%, what would happen to profit ?"
"If we closed down warehouses 2 and 5 and fed their customers
 from 3, how would this affect distribution cost ?"

Compound questions can usually be broken up into two or more
questions with the unrepeated conditions being common to them all.
"What was the temperature in Phoenix on Saturday at 12 noon and
at 4 p.m. ?"

BIBLIOGRAPHY

(1) McCarthy J., "Programs with Common Sense,"
Proc. Symposium on Mechanisation of Thought Processes,
H.M.S.O., London, 1959

(2) Phillips, A. V., "A Question-Answering Routine,"
Unpublished Masters Thesis,
Mathematics Department, M.I.T., Cambridge, Mass., 1960

(3) Green, Jr., B. F., et al., "Baseball: An Automatic
Question Answerer," Proc. WJCC, Vol. 19, 1961

(4) Simmons, R. F., et al., "Indexing and Dependency Logic
for Answering English Questions," American Documentation,
Vol. 15 1964

(5) Lindsay, R. K., "Inferential Memory as the Basis of
Machines that understand Natural Language,"
in "Computers and Thought," Feigenbaum, E. A., and
Feldman, J, eds., McGraw Hill, New York, 1963

(6) Raphael, B., "A Computer Program for Semantic Information
Retrieval," in reference (11) pp 33-145

(7) Woods, W. A., "Procedural Semantics for a
Question-Answering Machine.", Proceedings of the
FJCC 1968

(8) Woods, W. A., "An Experimental Parsing System For
    Transition Network Grammars." in "Natural Language
    Processing", Randall Rustin Ed., Algorithmics Press
    New York, N.Y. 1973

(9) Winograd T., "Understanding Natural Language,"
    Academic Press, New York, 1972

(10) Carbonell J. R., and Collins A. M., "Natural Semantics in
    Artificial Intelligence," Proc. IJCAI., 1973

(11) Minsky M., ed., "Semantic Information Processing,"
    M.I.T. Press, Cambridge Mass., 1968, pp 1-30

(12) Martin, W.A., "Translation of English into MAPL using
    Winograd's Syntax and a Semantic Case Grammer",
    Automatic Programming Memo 11
    Project Mac, M.I.T., April 17 1973

(13) Martin, W.A., "Things That Really Matter --
    a Theory of Prepositions, Semantic Cases and Semantic
    Type Checking,"
    Automatic Programming Memo 13
    Project Mac, M.I.T, August 15 1973

(14) Fillmore, C. T., "The Case for Case,"
    in Universals in Linguistic Theory, Bach and Harms ed.
    Holt, Rinehart and Winston, 1968

(15) Celce-Murcia, M., "Paradigms for Sentence Recognition,"
    in System Development Corporation final report
    No. HRT-15092/7907

(16) Martin, W. A., "MAPL 2,"

Automatic Programming Memo 12

Project Mac, M.I.T., August 1 1973

(17) Krumland R. B., "Concepts and Structures for a

Manager's Modelling System," Thesis Proposal,

Sloan School of Management, M.I.T., Oct 15 1973

[18] Martin, N. A., "Prog. 2."
Automatic Programming Team 12
Project Rec, M.I.T., August 1972

[19] Knowlton F. G., "Concepts and Structures for a
Manager's Modelling System," Thesis Proposal,
Sloan School of Management, M.I.T., Oct 20 1972