QUESTION ANSWERING VIA CANONICAL VERBS AND SEMANTIC MODELS: A MODEL OF TEXTUAL MEANING

Gary G. Hendrix

Technical Report NL 12

January 1973

Natural Language Research for CAI
Supported by
The National Science Foundation
Grant GJ 509X

The Department of Computer Sciences
and CAI Laboratory
The University of Texas
Austin, Texas 78712

ABSTRACT

A natural language question answering system is presented with special emphasis on the scheme used to model the meaning of a body of text. The principle components of this scheme are a state of the world graph (SWG) and a time line. The SWG encodes objects and relationships, as they appear in a brief instant, in a representation similar to that used by the STRIPS robot. The time line encodes the nature and sequence of events characterizing the changes in the state of the world with the passage of time. Mechanisms coupling the modeling scheme to natural language input and output are discussed and illustrated by examples.

Key Words and Phrases: Semantic model, question answerer, time frame, event scenario, canonical verb, semantic paradigm, robot.

QUESTION ANSWERING VIA CANONICAL VERBS AND SEMANTIC MODELS: A MODEL OF TEXTUAL MEANING

Introduction: The question answering system

One of the major obstacles confronting the builders of a natural language question answering system is the problem of adequately representing the meaning contained in the piece of text over which questions are asked. The goal of this report is to present a two pronged approach for surmounting this obstacle. First, English input sentences are parsed into a form of semantic net (7). In the net the event described by the English sentence is depicted by a canonical verb (6) and a set of event parameters associated with that verb. This parsing system, presented fully in a companion paper (10), provides the mechanism by which widely diverse surface structures with identical meaning may be transformed into a unique conceptual structure. Second, a STRIPS (3) robot-like model of the world is used to weave the sequence of events presented to the question answerer over several sentences into a unified whole.

A diagram of the overall system is shown in Figure 1. English statements or questions are given to the parser which produces a semantic net representation of the sentence employing canonical verb forms. The output from the parser is fed to an interface which determines what facts are to be recorded in the model or, in the case of a question, what information must be retrieved. The model acts as an information storehouse. After retrieval from the model, appropriate responses to input strings are fed to the generator from the interface. The generator (8,9) in turn produces English output.

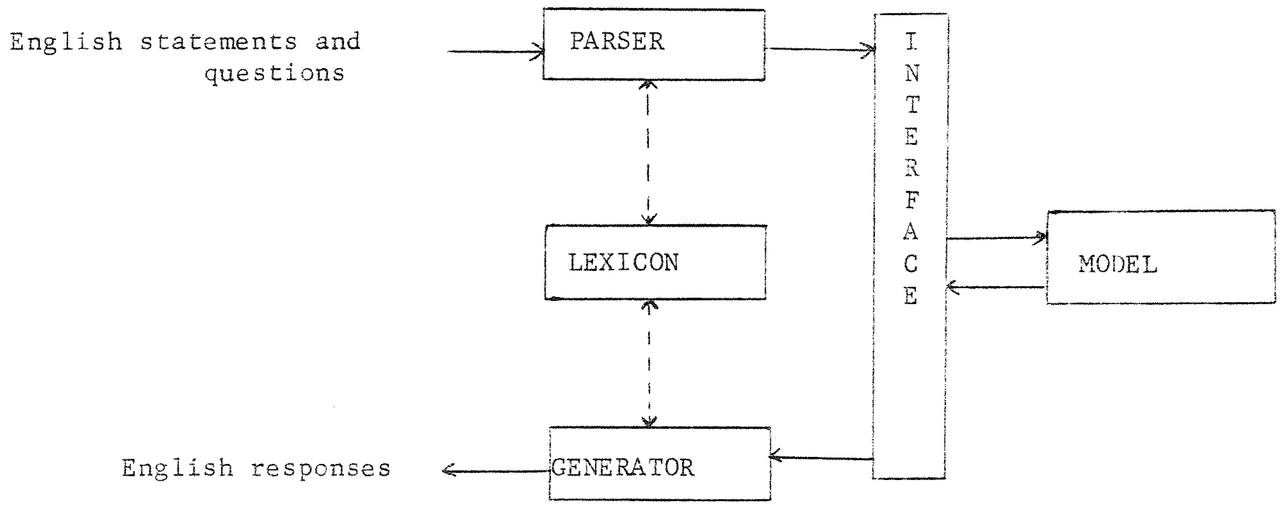


Diagram of Question Answering System

FIGURE 1

In practice, a chronological sequence of events constituting a story is given to the system as input. As the story unfolds, a model encoding the actors, props and events of the story is constructed. The system behaves as though it were recording the story on motion picture film.

The input sequence also may include questions which query portions of the story already told. To help answer such questions, the "film" produced by declarative sentences is reviewed.

Models and question answering

When asked a question, an intelligent entity has two alternatives for arriving at an answer. The entity may perform an experiment or it may use some theory or model. Many (perhaps all) questions about the past must be answered by models since it is impossible to actually turn back the clock to perform an experiment in the past. (The memory of a past experiment is a model, not an actual experiment.) By a similar argument, questions about the future must be answered by models if they are to be answered at all.

Experiments, of course, may be used to answer questions about the present. However, some use of a theory of model must be made to determine what set of experiments must be performed and how the results of the experiments are to be interpreted.

Hence, it is necessary to conclude that models in some form are necessary for most if not all question answering. The need for good models in answering questions by computer is particularly acute since the computer does not have (or at least is not often given) the ability to perform experiments in the real world.

Until recently, the use of semantic models has been only an implicit part of natural language systems. (According to the above, previous systems must have contained models, if only implicitly, since they lacked the ability to perform experiments.) More recent works such as Winograd's (11) and John Brown's (1) do use explicit models although their models are rather specific in nature. (i.e., Winograd's model is confined to the block's world and Brown's to some well defined physical systems.)

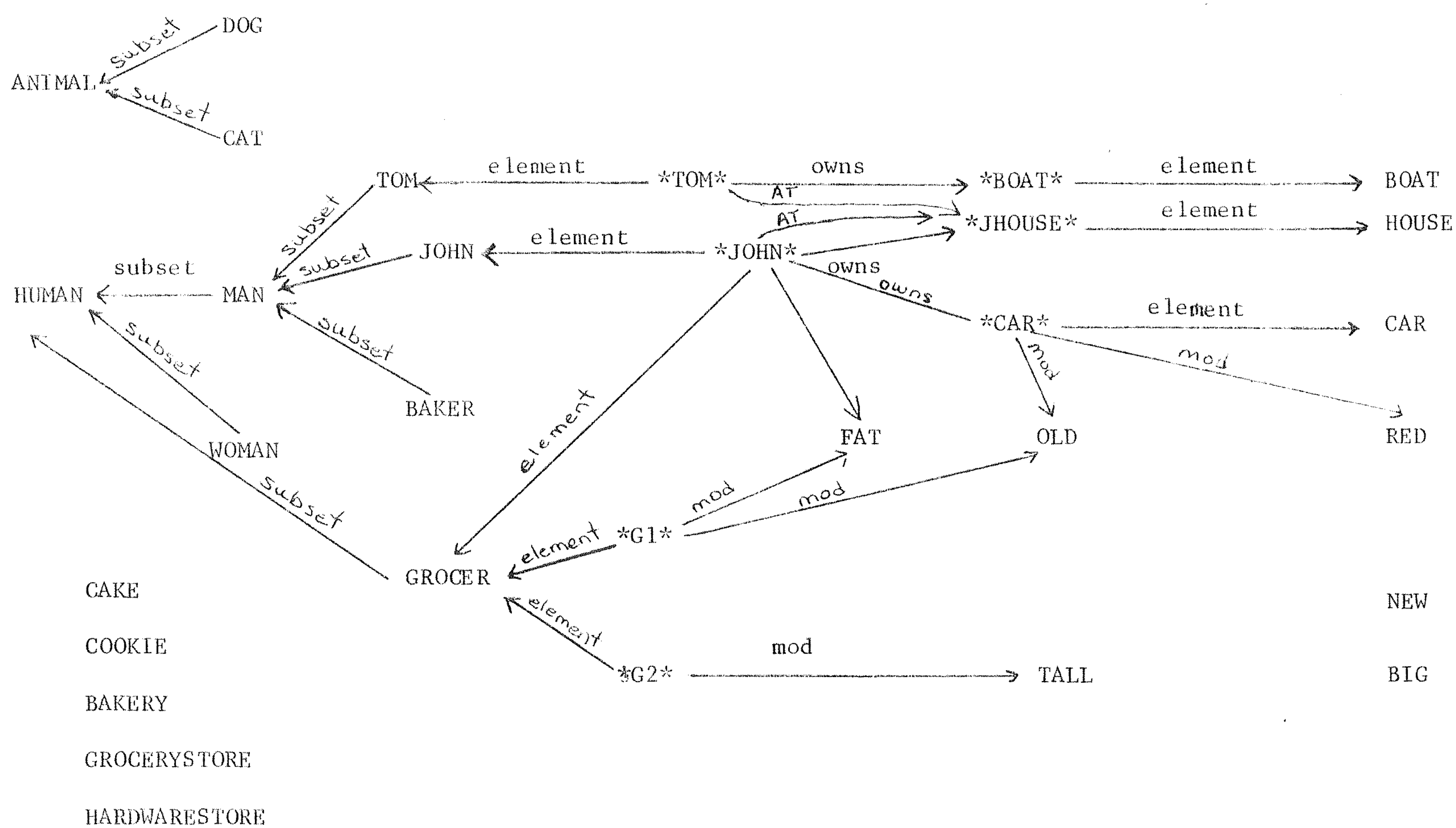
The Modeling system

The purpose of the modeling portion of the question answering system is to encode a collection of facts or concepts. Three distinct types of concepts must be encoded:

1) Objects
2) Relationships between objects
3) Changes in relationships with respect to time

The modeling system uses an encoding procedure analogous to that used in motion pictures. Motion picture film is actually a sequence of still photos. Objects are recorded in the still photographs by their images. Physical relationships between objects are recorded by the physical arrangement of images in the still photos. Changes in relationships with respect to time are recorded by the sequence of still photographs.

The "still photos" of the modeling system are graphs representing the state of the world at an instant in time. Nodes in the graph stand for objects, and arcs represent (binary) relationships between objects. No time relationships are included since time is effectively eliminated as a variable by considering only a brief instant in each graph. Figure 2



A State of the World Graph

FIGURE 2

is an example of a state of the world graph (SWG). Node names beginning and ending with asterisks represent names which have been GENSYMed by the system. Other node names are words in the system's lexicon.

Consider the node labeled *JOHN*. *JOHN* is an element of JOHN and GROCER. JOHN is the set of all Johns, a subset of MAN, the set of all men. GROCER is the set of all grocers. Hence, *JOHN* is a node representing a particular man who is a John who is a grocer. This John owns a particular house, *JHOUSE*. Further, John is at that house, owns an old, red car and is fat.

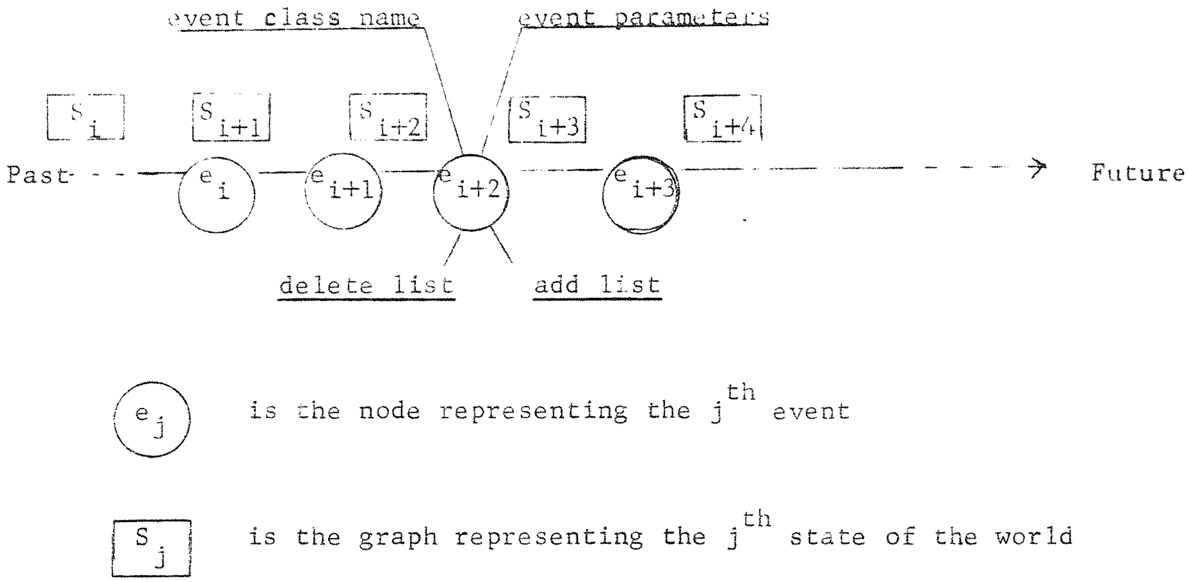
To model changes in relationships with respect to time, a sequence of SWGs, analogous to the sequence of still photographs of motion pictures, is defined. The sequence of events causing changes in the state of the world is recorded on a time line (see Figure 3). Each node on the time line corresponds to an event. Beginning with some initial state, the state of the world following any event may be determined by considering the sequence of events along the time line. Associated with each event are a list of relationships which held before the event took place but which are no longer valid after the event (the delete list) and a list of relationships which did not hold (or at least were not known to hold) before the event took place but which are valid after the event (the add list). Thus, each node on the time line models a set of changes in the state of the world produced by the occurrence of some event. To specify the event causing the changes, an event class name and a list of parameters are also associated with each time line node.

When the question answering system first begins to operate, it begins with a SWG representing conditions before the first event. As descriptions of events are given as inputs, new nodes are added to the time line. With each new event, certain relations (those on the event's delete list) are deleted from the current SWG and new relationships (those on the add list) are added. Thus, the old picture of the state of the world is transformed into a new picture showing conditions after the event. To answer questions about relationships in the past, the movie may be run backwards by considering the sequence of events in reverse order and deleting the adds and adding the deletes. Once a question is answered, an analogous forward process resets the SWG to reflect current conditions. Thus, only one SWG need be kept.

Questions regarding the nature of events themselves (as opposed to questions concerning relationships which are altered by events) are answered by consulting the record of events on the time line.

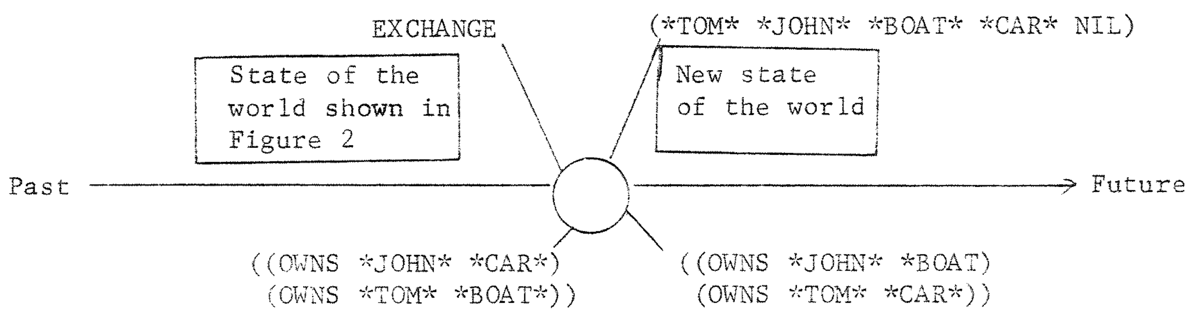
Most ordinary events may be specified by a variety of verbs. For example, the event of buying may also be specified as a selling or a trading. Clearly, "John sold the car to Tom" means the same as "Tom bought the car from John." To simplify the modeling system, each class of events is allowed to have only one name. Hence, the verbs "buy", "sell", "trade", etc., are all recorded under the event name EXCHANGE. That is, any node on the time line modeling a buying or a selling will have associated with it the event class name EXCHANGE. The transformation of various English verbs into the canonical form used in the modeling system is performed by the parser section of the question answering system described in the companion paper.

An event is not adequately specified by its name alone. A list of event parameters is also needed. For example, EXCHANGE means very little unless it is accompanied by a list of parameters specifying the



Abstract Diagram of the Time Line

FIGURE 3



Time Line Record of an EXCHANGE Event

FIGURE 4

seller, the buyer, the thing bought, etc. Clearly, not all the parameters need be known ("John bought something at the bakery"), but the more parameters which are known, the better the event is specified. Hence, with each time line node there is associated a list of parameters the length and order of which is determined by the associated event class.

As an example of a time line node, consider how the event "John traded his car to Tom for Tom's boat" would change the SWG of Figure 2. The relationships (OWNS *JOHN* *CAR*) and (OWNS *TOM* *BOAT*) would no longer be true. (OWNS *JOHN* *BOAT*) and (OWNS *TOM* *CAR*) would become true. Hence, a time line node like the one shown in Figure 4 would be produced.

Sometimes the cause of a change in the state of the world is not known. In such instances the event class and parameter lists associated with the time line node are marked NIL.

A basic manipulation function

The basic operations used to manipulate the model's data base is ADVFRM (advance frame). ADVFRM is a function of four arguments, the name of an event class, the parameters associated with the event, a list of relationships which are to be deleted from the SWG as a result of the event, and a list of relationships which are to be added. ADVFRM creates a new node on the time line and associates the arguments with that node. Further, for each relation (r e_1 e_2) on the delete list, ADVFRM removes the arc r which joins the nodes e_1 and e_2 in the SWG, if such an arc exists. If such an arc does not exist, (r e_1 e_2) is deleted from the delete list associated with the time line node. (This prevents the arc from being erroneously created when the model is backed up.) For each relation ($r\ e_1\ e_2$) on the add list, nodes with labels e_1 and e_2 are created if need be and an arc labeled r is created to join them. (If such an arc already exists, ($r\ e_1\ e_2$) is deleted from the add list associated with the time line node.)

If the first two arguments of ADVFRM are NIL, then no reason is given for the change in the state of the world. By use of this feature, initial conditions may be set up.

Events as operators

Since an event transforms one state of the world into another, it is reasonable to think of events as operators. Pursuing this notion, an event class name becomes the name of an operation which transforms the state of the world (an implicit parameter) and a set of explicit event parameters into a new state of the world. In terms of the model, the event class name becomes the name of a procedure (an <u>event procedure</u>) which transforms the SWG and a list of explicit parameter values into a new SWG.

Such event procedures with all the power of Turing machines could be written for each verb class. However, in the present work a much more restricted view of these event procedures has been taken, paralleling the definitions of operators for the STRIPS robot. In the STRIPS robot, an operation is defined in terms of a set of preconditions (a list of relationships) which must hold in the state of the world before the operation may be applied, and both a list of relationships which must be deleted as a result of the operation and a list of relationships which must be added.

Several features of the STRIPS robot operators appear to be immediately applicable to event procedures. The operators are called by the operation name and a list of explicit parameters with the STRIPS state of the world model as an implicit parameter. In parallel, event procedures are to be called by the event class name and a list of explicit event parameters with the SWG as an implicit parameter. Further, there is an unmistakable correspondence between the add and delete lists of the STRIPS robot and the add and delete lists which are associated with time line nodes. Thus, the only component of a STRIPS operator which does not appear immediately useful is the precondition list.

The precondition lists of operators are used by the STRIPS robot in determining a workable sequence of operations for achieving the robot's goal. In contrast, a question answerer which receives a chronological accounting of events has no need to determine the event sequence since it is given. There is, however, an analogous problem which may be seen in the following example. Suppose that part of the input to the question answerer is "John bought a clock at the hardware store. Then John bought a cake at the new bakery." Note that John was at the hardware store when he bought the clock. The next event has him buying a cake at the new bakery. Before the event at the bakery could take place (as a precondition to the bakery event), John must have gone (perhaps by a complex route) from the hardware store to the bakery. Thus, the necessity of satisfying preconditions clearly remains a problem even when the event sequence is given. In the question answering system, the precondition problem is the problem of determining the nature of unspecified intermediate events which set the stage for the accomplishment of specified events.

The nature of these unspecified intermediate events may, to a large extent, be determined by the preconditions of the specified events. Returning to the example, before "John bought a cake at the new bakery" some event must have occurred which deleted the relation that he was at the hardware store and added the relation that he was at the bakery. In general, EXCHANGE events must be preceded by events which bring the participants in the event to the place of exchange, etc.

In the place of the preconditions used by STRIPS robot operators, an event procedure substitutes the add and delete lists of an unspecified intermediate event which transforms any state of the world into a state satisfying the preconditions of the main event. That such an event must have occurred is implied by the sequence of specified events. Thus, a call to an event procedure causes two events to be incorporated into the model, creating two nodes on the time line.

An example of an event procedure, a simplified version of EXCHANGE, is shown below.

<u>procedure call</u>

EXCHANGE(seller buyer thingbought thinggiveninexch place)

<u>intermediate delete list</u>

((AT seller *) (AT buyer *) (AT thingbought *) (AT thinggiveninexch *)
(OWNS * thingbought) (OWNS * thinggiveninexch))

<u>intermediate add list</u>

((AT seller place) (AT buyer place) (AT thingbought place)
(AT thinggiveninexch place) (OWNS seller thingbought)
(OWNS buyer thinggiveninexch))

<u>main delete list</u>

((OWNS buyer thinggiveninexch) (OWNS seller thingbought))

<u>main add list</u>

((OWNS buyer thingbought) (OWNS seller thinggiveninexch))

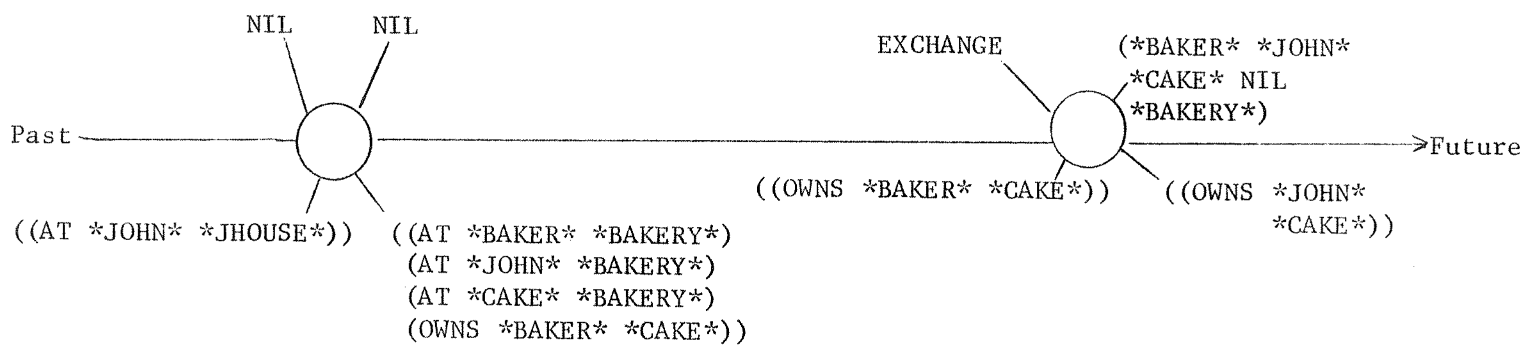
Consider the actions performed by the event procedure EXCHANGE if the system is presented with the event "At the new bakery John bought a cake from the baker" while in the state shown in Figure 2. By a mechanism to be discussed shortly, the system finds the SWG node *JOHN* which represents the particular John under discussion and creates nodes *BAKERY*, *CAKE* and *BAKER* to represent the bakery, the cake and the baker. Relationships integrating these new nodes into the original graph are also produced. With these nodes (node names) as arguments, procedure EXCHANGE is entered by the call

EXCHANGE(*BAKER* *JOHN* *CAKE* NIL *BAKERY*)

where NIL represents the unspecified parameter thinggiveninexch. The effect of the call on the time line is shown in Figure 5. Stepping through the procedure:

1) A time line node is created for the intermediate event with event class name and parameter list set to NIL.

2) Working down the intermediate delete list of the event procedure (AT seller *) is considered first. An attempt is made to match this relationship against those encoded in the SWG. The asterisk is allowed to match anything, but "seller" is bound to *BAKER*. Thus a matching relation must be of the form (AT *BAKER* --). Since no such relation exists in the SWG, no action is taken. The next relationship on the delete list is (AT buyer *) which matches (AT *JOHN* *JHOUSE*) in the SWG. Hence, (AT *JOHN* *JHOUSE*) is deleted from the SWG and the relation



<u>Time Line Nodes Created by an Event Procedure</u>

<u>FIGURE 5</u>

put on the delete list of the time line node. The relationships (AT thingbought *), (AT thinggiveninexch *), (OWNS * thingbought) and (OWNS * thinggiveninexch) match nothing in the SWG and hence cause no action.

3) Working down the intermediate add list, (AT seller place) causes (AT *BAKER* *BAKERY*) to be added both to the SWG and to the intermediate time line node add list. Likewise, (AT *JOHN* *BAKERY*) and (AT *CAKE* *BAKERY*) are added. Since thinggiveninexch is bound to NIL, (AT thinggiveninexch place) causes no action. Proceeding down the add list, (OWNS *BAKER* *CAKE*) is added while the relation (OWNS buyer thinggiveninexch) causes no action.

4) A time line node is set up for the main event with the event class name set to EXCHANGE and the parameter list set to (*BAKER* *JOHN* *CAKE* NIL *BAKERY*).

5) By working down the main delete list of the event procedure, (OWNS *BAKER* *CAKE*) is deleted.

6) By working down the add list (OWNS *JOHN* *CAKE*) is added.

Steps 1 - 3 and 4 - 6 are accomplished by calls to ADVFRM.

By this procedure the original state of the world (as shown by the model) is transformed into an intermediate state in which John and the baker are at the bakery and the baker owns the cake. This intermediate state is then transformed into a state in which John and the baker are still at the bakery, but John owns the cake. The reader is encouraged to step through the EXCHANGE procedure starting with various states of the world and using variations of the event parameters.

Preparing English input for use in the modeling system

Left unexplained in the last section was the mechanism by which English sentences are transformed into event procedure names and lists of explicit parameters naming nodes in the SWG. Rather than define this rather subtle mechanism in detail, an example showing how the mechanism transforms indicative affirmative sentences will be presented. (The mechanism differs somewhat for interrogative sentences.)

The input sentence

"At the new bakery John bought a cake from the baker"

is transformed by the parser into a semantic net. A simplified representation of that net is as follows.

```
(SENTENCE_x  (CANONICAL-VERB-FORM EXCHANGE)
             (SELLER     (PRINTIMAGE BAKER DETERMINER DEFINITE))
             (BUYER      (PRINTIMAGE JOHN DETERMINER DEFINITE))
             (THINGBOUGHT  (PRINTIMAGE CAKE DETERMINER INDEFINITE))
             (PLACE       (PRINTIMAGE BAKERY DETERMINER DEFINITE
                           MOD (PRINTIMAGE NEW DEGREE POSITIVE) ))
             (MODAL       (PAST INDICATIVE AFFIRMATIVE)) )
```

The procedure name, EXCHANGE, is indicated as the CANONICAL-VERB-FORM. Information relating to event parameters is contained in property lists preceded by such parameter names as SELLER, BUYER, etc. Any parameter not defined by a property list in the net structure is set of NIL.

Each parameter defined explicitly by the parser output must be bound to the name of some node in the SWG. If no appropriate node exists in the SWG, one must be created. To do this a routine, FOC (find or create), is called with a parameter's property list as its argument. If the parameter's determiner is indefinite, then FOC simply creates a new node in the SWG satisfying the other properties on the property list. Thus, when "a cake" is mentioned, FOC creates a new cake and does not concern itself with whether or not this new cake is some cake already modeled in the system. (Should it later be determined that two nodes actually model the same entity, a special COLLAPSE function is used to merge them.) If the parameter's determiner is definite, then FOC attempts to find an existing node in the graph satisfying the description of the noun. Thus, when FOC is to return the name of a node corresponding to "the new bakery" it assumes that a new bakery has been talked about before and it looks for a node in the SWG which could be modeling that bakery. If more than one such node is found, the last one mentioned is returned. (Each time a node is used a use-time is associated with it.) If no such node is found, one is created.

The value of the attribute PRINTIMAGE on a parameter's property list is assumed to be the name of a node representing a set of entities of which the value of the parameter is an element. In the case of the parameter SELLER, BAKER is the name of a set of which the value of SELLER must be a member. Words such as BAKER, JOHN, CAKE, etc., are entered in both the system's lexicon (for use by the parser) and the SWG before processing begins. Certain primitive relationships among the sets named by these words are also preset. For example, the relation (SUBSET JOHN MAN) is preset in the SWG. Thus, whenever a John is mentioned he becomes an element of JOHN, a subset of MAN, and is therefore known to be a man.

The parameters for "At the new bakery John bought a cake from the baker" are processed from the semantic net as follows.

1) The SELLER is determined. FOC looks for an x in the SWG such that (ELEMENT x BAKER). Finding no such x, one is created (call it *BAKER*) and it becomes the value of SELLER. (During this process (ELEMENT *BAKER* BAKER) is encoded in the SWG.)

2) The BUYER is determined. FOC looks for an x such that (ELEMENT x JOHN). It finds *JOHN* which becomes the value of BUYER.

3) THINGBOUGHT is determined. Since the determiner is indefinite, FOC creates a new node x such that (ELEMENT x CAKE). Assume x is named *CAKE*.

4) PLACE is determined. In this instance the job of FOC is complicated by the presence of the modification. FOC looks for an x such that (ELEMENT x BAKERY) and (MOD x NEW). Finding none, FOC creates such an x and makes it the value of PLACE.

Once the values of parameters have been determined, the system calls EXCHANGE to encode the event. It is important to note that relationships encoded in the SWG during the process of defining new nodes to serve as parameters are not entered on the add of delete lists of the event's time line nodes. Thus, the new nodes are not eradicated if the model's time frame is backed up over the event. For example, the model will always know who the baker is.

Processing interrogative sentences

The processing of interrogative sentences closely parallels that of declarative sentences so far as the determination of parameters is concerned. Of course, rather than direct the system to change the SWG and extend the time line, questions cause the SWG and time line to be examined. There are three basic types of questions the system is capable of accommodating.

An example of the first type is "What man bought a cake at the bakery?" Questions of this type are answered by examination of the time line. For the current example, a search down the time line (toward the past) is conducted until an event is found whose event class is EXCHANGE and whose PLACE parameter has the value *BAKERY*. When such a node is found a test is made to see if the value of the THINGBOUGHT parameter is an element of CAKE. This test being passed another test is performed to determine if the value of BUYER is an element of MAN. If this test is passed then the value of BUYER is the answer to the question. This answer is used to create a property list for the parameter BUYER in the semantic net which posed the question. After some slight alterations, this semantic net is given to the generator which produces any one of a number of paraphrases of "John bought a cake at the bakery."

An example of the second type of question is "What did the baker own before John bought something at the bakery?" The parser splits the question into two parts. A search similar to the one just described is used to find a node on the time line corresponding to the event "John bought something at the bakery." The SWG is then backed up to just before that event and the system investigates the SWG for an x such that (OWNS *BAKER* x). If such an x is found, it is the answer to the question.

An example of the third type of question is "What did the baker own before John owned a cake?" To answer this question the SWG is stepped back until a state is found in which (OWNS *JOHN* x) and (ELEMENT x CAKE) are true. Then the SWG is stepped back until one of these relationships is no longer true. Finally, the SWG is stepped back until (OWNS *BAKER* z) is true for some z. If such a z can be found, it is the answer to the question.

Implementation

The semantic modeling system and other parts of the question answer have been implemented in GROPE, a graph processing language embedded in FORTRAN, on the CDC 6600 at the University of Texas. A number of experiments have been conducted using various versions of event procedures for the event classes EXCHANGE and GO-CARRY. (A special procedure was also written for BE. Since BE does not express action, it causes no nodes to be added to the time line but simply changes the SWG. Sentences such as "John is at Tom's house", "John is a grocer", and "John owns the car", are all handled through BE by various mechanisms.)

The system, while conceptually simply and easily programmable, has proven to be quite satisfactory in answering questions relating to sequences of events. Indeed, for long sequences which follow no particular pattern, the system is almost always better able to keep track of the order of events and the complete current state of the world than a human competitor. Response time for most questions is in the neighborhood of one or two seconds.

An interesting difficulty posed by the definition of the event procedure EXCHANGE was discovered when the system was given the input "John bought a clock at the hardware store. Then John bought a cake at the new bakery." When asked the question "Where is John's clock?", the system answered "John's clock is at the hardware store." A human question answerer would probably assume that John carried the clock with him and would answer "At the bakery." Had John gone from the hardware store to his home and then to the bakery, a human would probably assume that John's clock was at John's house.

Suppose the system knows that Tom owns a boat, a house and $100. If the system is first told "Tom traded something for John's car", and then asked "What does Tom own?", the system replies that Tom owns a boat, a house, a car and $100. Clearly, Tom does own the car, but the boat, house and $100 are all questionable. The SWG needs to be generalized to indicate which information is known to be factual and which is merely speculative. Such a generalization might also help cure the problem of the preceding paragraph.

As a final criticism, the strict chronological sequence demanded by the system causes input texts to make very boring reading. Rather than being restricted to time line growth on the right, event procedures using tense (2) and other time clues provided by the input sentences might be able to insert events into the history portion of the time line. Even with such an extension, the system would still be unable to account for the simultaneous occurrence of multiple events or the occurrence of events in an unknown order. Both of the latter problems could conceivably be solved by generalizing the time line to a directed graph.

Acknowledgements

I would like to acknowledge the advice and guidance which I received from coworkers on the question answering system project. Many of the ideas presented in this paper were molded through interactions with Jonathan

Slocum, Craig Thompson, Henry Armus, Jr., and with our professor, Dr. Robert Simmons.

REFERENCES

1. Brown, John S., R. B. Burton and Frank Zdybel, "A Model Driven Question-Answering System for a CAI Environment," University of California, Irvine, Department of Information Science, Technical Report #13, January 1972.

2. Bruce, Bertram C., "A Model for Temporal References and Its Application in a Question Answering Program," Artificial Intelligence, III, 1972, 1-25.

3. Fikes, Richard E. and Nils J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," Artificial Intelligence, II, 1971, 189-208.

4. Lindsay, P. H. and D. A. Norman, Human Information Processing, Academic Press, New York, 1972.

5. Minsky, Marvin L., "Matter, Mind, and Models," In Minsky, Marvin L. (Ed.), Semantic Information Processing, MIT Press, Cambridge, Mass., 1968.

6. Schank, Roger, "Identification of Conceptualizations Underlying Natural Language," In Schank, R. and K. Colby (Eds.) Computer Cognition, Freeman Press, (In Press).

7. Simmons, R. F., "Semantic Networks: Their Computation and Use for Understanding English Sentences," University of Texas, Austin, Department of Computer Sciences, Technical Report #NL-6, May 1972.

8. Simmons, R. F., and J. Slocum, "Generating English Discourse from Semantic Networks," CACM, XV, 1972, 891-905.

9. Slocum, Jonathan, "Question Answering via Canonical Verbs and Semantic Models: Generating English from the Model," The University of Texas, Austin, Technical Report #NL-13, January 1973.

10. Thompson, Craig W., "Question Answering via Canonical Verbs and Semantic Models: Parsing to Canonical Verb Forms," The University of Texas, Austin, Technical Report #NL-11, January 1973.

11. Winograd, T., Understanding Natural Language, Academic Press, New York, 1972.