ACTIVE SEMANTIC NETWORKS AS A MODEL OF HUMAN MEMORY[1]

David E. Rumelhart
Donald A. Norman
Department of Psychology
University of California, San Diego
La Jolla, California 92037

Abstract

A general system to simulate human cognitive processes is described. The four-part system comprises a nodespace to store the network structure; a supervisor; a transition network parser; and an interpreter. The method by which noun phrases operate and the process for the determiner "the" is presented. An analysis of verb structures illustrates how network structures can be constructed from primitive verb definitions that get at the underlying structures of particular verbs. The paper concludes with an illustration of a problem in question-asking.

A Model of Human Memory

We have constructed a large general simulation of human language and long-term memory on the premise that the study of the inter-relationships among psychological processes will lead to more insight into human cognition and memory. The general implementation is basically complete, and a variety of users are starting to study specific psychological tasks (language understanding; children's development of language; primitive verb structure; reading; inference; game playing--Go and Gomoku; visual representation and memory; learning; and question answering). It is still *too* early *to* report on the results of the psychological investigation.. Therefore, this paper is a progress report on the system and the underlying psychological principles.

The major guidelines have come from our attempts to represent long-term memory structures. We know that people rapidly forget the details about the surface structure of an experience but retain the meaning or interpretation of that experience indefinitely. We also know that retrieval of an experience from memory is usually a reconstruction which is heavily biased by the person's general knowledge of the world. Thus, general world knowledge should interact with specific event knowledge in such a way that distinction between the two is not possible. The representation should allow paraphrase. Finally, the limitations of human working storage (or short-term memory) probably comprise a fundamental property of the system, one that should be viewed as an essential, positive component, not as simply a performance limitation.

The Computer System

The basic system consists essentially of four fixed components: 1) a nodespace in which our network structures are stored; 2) a supervisor which allows us direct access to various portions of the nodespace; 3) a parser which converts strings of words into network structures; 4) an interpreter which processes sections of the nodespace and carries out any strategies which were stored in that portion of the nodespace. The system is written in ALGOL on the Burroughs 6700 at the university of California, San Diego. The simulations are done in our own English-like language, with all statements entered through the parser. The language is called SOL (for Semantic Operating Language—pronounced "soul") and it is specifically designed for manipulating and traversing the network structures of the data base. Because we wish many different psychological simulations to be handled by the one system, we have made it reasonably general and readily extendable so that any of the psychological hypotheses under study can be simulated and tested in its own specialized mini-world.

The Representation of Actions and Concepts. The representation to be described here is presented in more detail and with more justification in the papers by Rumelhart, Lindsay & Norman[3] and Norman[4]. Basically, we use a network representation with nodes connected to other nodes by labeled, directed relations. Because each relation also has an inverse, the network is bi-directional.

Events are specified in a similar way, except that actions require arguments. Thus, the node that represents an action may have obligatory relations leading from it, specifying such things as the agent, location, and object of that action.

Most actions and concepts in the network have a single primary node (or type node) that encodes its definition, and numerous secondary nodes (or token nodes) that represent specific instances of the primary one. Almost all encodings of specific scenes are done by means of secondary nodes.

The basic unit in the memory space is the scenario: an action that consists of events, agents, locations, and objects. To illustrate the representational system, consider the sentence

Peter put the package on the table.

Figure 1 shows a possible simple encoding for this sentence which includes some of the underlying structures of the action.

Figure 1. Peter put the package on the table.

The SOL Language

The parsing process is based on three independently motivated principles. First, the parsing procedures are represented as an augmented recursive transition network (following the work of Woods and Kaplan[5,6,7]). Second, the parser is based around a "case grammar" (after Fillmore[8]) and has "case frames" and "argument constraints" associated with many lexical items. (Here some of the methods suggested by Schank[9] can be used.) Third, the parsing is based on the idea that it is the task of each noun phrase to find its own referent in memory if it exists or else to create a new structure in the data base. Thus, certain lexical items such as determiners, adjectives, and pronouns are defined by the strategies for finding the proper referent.

Argument Frames. Associated with every predicate word is an argument frame which indicates which and how many arguments must exist. For example, associated with the verb move might be the following set of arguments; 1) a causal mover (called here an AGENT); 2) a moved object (OBJ); 3) an initial location (FROM-LOC); 4) a terminal location (T0-L0C); 5) a means of moving (METHOD); and 6) a time of occurrence (AT-TIME). We denote the argument frame as follows:

AGENT X MOVES Y (FR0M-L0C LI T0-L0C
L2 METHOD M AT-TIME T).

Those arguments enclosed in parentheses are taken to be optional; the others are required. Associated with each case name (e.g., FROM-LOC or METHOD) is a list of prepositions which can occur at the surface level to indicate or mark that argument. Each label also is

associated with a set of semantic characteristics which can be interrogated during the parse. The prepositions and the semantic characteristics can be used together to disambiguate which of the variety of concepts a given noun phrase is representing.

Certain verbs, particularly those talking about ideas, sometimes take whole sentences as arguments. Such arguments are referred to in our system as propositional arguments (PROPOSITION). Thus, the argument frame for one sense of the verb make (as in the sentence "Freddy made his brother come home") takes a propositional argument and has the argument frame

AGENT X MAKE PROPOSITION Y (METHOD M AT-TIME T)

where Y stands for some transformed version of an entire sentence.

At every -point during the parse the goal is to find and correctly fill the argument slots of the predicate word in question. If some arguments do not fit into the frame of the sense of the predicate word in question, a new sense of the predicate word is tried until either a fit occurs, or no more senses exist (in which case, *the parse* fails).

Operators. One important class of words in our language analysis is the class we call operators. Operators are nouns that take arguments (usually prepositional phrases) and thus have associated case frames. Operators can be verb based nouns such as destruction in the destruction of the city by the enemy—destruction is an operator with its two arguments filled by the following noun phrases. An operator is also a relational noun such as father, as in the sentence "Bill is the father of Henry." Here, father is analyzed as an operator with one argument. The existence of case frames for these nouns as well as verbs reduces substantially the ambiguity of prepositional modification.

Disambiguating the Referent

One of the major problems in the analysis of natural language is determining the exact referents of a phrase. Most of the complexities of such words as the come from the difficulties of determining just what concept is being referred to. In the SOL system the parser automatically invokes the procedural definition of the which, in turn, performs an active search through the data base to determine the referent as each noun phrase is analyzed. We illustrate here how this is done by going through the strategies that comprise the procedural definition for the. In rough form, the process is this: first, if the phrase is an operator, then it contains the procedures for its own disambiguation which should be performed before doing anything else. If that is not the case, then we determine whether the object being referred to is unique within the data base, for if it is, no particular problem exists. If these two strategies fail, then we see whether or not immediate context helps, and if that fails, we look to see whether or not there is a relative clause that can do the job. Now look at this in detail.

Operators. If the unknown phrase is an operator, then it is necessary to determine whether or not to perform the operation or to refer to the value of the operation. Thus, with the phrase the father of John *the* operator father has not been evaluated, *so* first we execute the routine for father (passing John to it as an argument) and then return to the parser with the result of that operation (presumably, the name of the person who is John's father). If father is being used in its nominal sense, however, as in "I told the father to give the toothbrush to the daughter," then we are referring to the value that a previous execution of the operator had returned.

Unique Instances. If a given concept *is* unique *to* the data base, then it can be unambiguously found whenever referred to with a determiner. Thus, if the memory system knows of only one ocean, to tell it "The sun set over the ocean" is completely unambiguous, not because the system is intelligent, but rather because it doesn't know enough to be confused. Tell it about the existence of a second ocean (or a second sun) and this strategy will not work (but the following ones might).

Foregrounding. Chafe[1]" suggests that many problems in disambiguation are handled by context in a manner that he calls "foregrounding." If the recent context has been about "Fred's kitchen," then the objects in that particular kitchen are foregrounded even though they have never been mentioned specifically. Foreground establishes local context. In our system each concept that can be brought to the foreground has associated with it a specific list of items. As new sentences pass through the parser, they initiate the appropriate foreground lists.

Note that foreground has several hierarchical levels, for the context includes the general overall topic under discussion, the specific details, and the environmental setting of the speakers. Thus, in this paper we could now talk of "this conference" or "this parser," both of which would be disambiguated by foreground-like operations, but each would be at different levels.

Short-term Memory. We can also look back in short-term memory to determine if any of the recent sentences help disambiguate the referent. At the moment, we look back over the last five sentences. Eventually, we intend to have a more reasonable simulation of human short-term memory processes, so that only topics that could reasonably be expected still to remain in active short-term memory could be disambiguated this way.

Search. If all this fails, it is still possible that an intelligent search among the concepts discussed recently (or foregrounded *recently) could* disambiguate the referent. This strategy has not yet been implemented, primarily because its use depends upon the operation of a search routine that is not yet fully operational. (The search routine is a simultaneous breadth-first search emanating from as many nodes as are specified, returning with a path that links all the nodes in the search space. That path is evaluated for its logical properties and the search process is either terminated or continued.)

Clauses. A common method of disambiguation is by the use of clauses, as in the phrase the girl (whom) I saw in the park. This method of disambiguation is clearly an important part of normal English, It has been deleted from the existing the routines because the search routines do not yet work. But it is an important enough process to warrant further discussion here.

Consider the sentence "I see the girl with the telescope." As it now stands the sentence is incomplete and, therefore, ambiguous: we need some context. Suppose that the following information is known by the system.

Jane, Mary, Cynthia, and Helen are girls.
Mary has a telescope.

These data are represented in the left part of Figure 2.

Figure 2.

The analysis of the sentence "I see the girl with the telescope" is simple until we reach the phrase the girl. Thus, we can recognize I as the subject of the verb see. (The model has only one person with whom it converses, namely you. The change in designation of the subject to the case relation of agent occurs with the construction of the deep parse and construction of a permanent memory segment.) The analysis of the is complex because all the strategies discussed so far would fail. We need to look at the clause with the

telescope. A search of the data base reveals that only one girl possesses a telescope; now we have disambiguated the referent (see Figure 2).

A different result would occur had the contextual information in the data base been the following.

Mary is a girl.
I got a telescope on Tuesday.

The resulting analysis is shown in Figure 3.

Figure 3.

The major difference between the analyses shown in Figures 2 and 3 is that in the latter the phrase with the telescope is neither needed to help disambiguate the referent for the girl nor is it consistent with the known information about Mary. Hence, the referent program completes its action with one phrase left unanaly2ed. When control returns to the parser, this phrase is still left. The parser then checks it against the possible frame for the verb see and, in this case, finds that it can be used as the instrument of seeing. Again, the sentence is analyzed with no difficulty and with no recognition by the parser that an alternative analysis was possible.

Defining Verbs

At this point the general description of the system is complete. One more specific point is appropriate to discuss here, however. The basic premise underlying the linguistic analysis is that we can represent the meaning of verbs as network structures built from a limited set of semantic primitives. Here we wish to illustrate one analysis *of* verbs and 'their underlying primitives, both to show how we believe the linguistic structures should be represented and also to demonstrate several features of the SOL language.

At least three different aspects of verb meanings can be distinguished: states; changes of states; and causes of these changes. The stative component of a verb conveys that fixed relationship which holds among its arguments for a specified period of time. The change component of a verb tells that a change in state has occurred. The causative component communicates the source of, or reason for, the change. These different verb components are not all present in all verbs, but all components may appear in a single lexical item.

In the remainder of this section we show how we represent these various semantic components and how we can express the definitions of particular lexical items in such a way that the primitive representation for that item is automatically computed whenever it appears in a sentence.[11]

Statiyes. The simplest semantic component of verbs is the stative component. This component merely communicates the information that a particular state of the world holds from some initial time to some final time. The simple locative is an example of a verb which seems to have only stative components. For example;

A stadium was located in the park from 1956 until 1963. (1)

Sentence (1) presumably communicates nothing more than that a particular relationship held between a stadium and a park for some period of time. We represent this meaning by an underlying locative primitive called *LOC (the names of our primitive predicates are preceded with asterisks in order to differentiate them from surface lexical items). Figure 4 illustrates the network representation we give to sentence (1).

Figure 4,

We want to define *LOC and locate in such a way that when the meaning of locate is computed (i.e., the definition is executed), we have the structure given in Figure 4 generated in the nodespace and associated with sentence (1)- To accomplish that, we first define *LOC so that it generates the appropriate structure. Then we define locate in terms of *LOC. First the definition of *LOC:

```
Define as predicate *LOC.
X *LOC AT-LOC L (FROM-TIME T1 TO-TIME T2).
Return with newtoken for "*LOC" "SUBJ" X
   "AT-LOC" L "FROM-TIME" TI "TO-TIME" T2.
```

In this definition, the initial line calls the special defining mode of the parser which sets up the basic node structure for the definition of a new concept. It also accepts the sentences that follow as instructions for processes which are executed each time the newly defined structure is actually used. The term predicate is the syntactic class to which *LOC is being assigned. This class includes all relational terms which can stand as the main relational term of a sentence. The second line of the definition gives the argument frame for the definition. In this example, the structure that *L0C returns is a newly constructed token node (secondary node) for the primitive with the appropriate argument values inserted in place.

Now we can define the stative sense of the verb locate:

```
Define as predicate LOCATE.
X LOCATE AT-LOC L (PROM-TIME TI TO-TIME T2).
Iswhen X *LOC at L from T1 to T2.
```

(Other senses of locate can also be defined, but they are not shown in this example.) Note here that when the definition of locate is invoked, a statement involving *LOC is asserted. Whenever this happens, the definition of *LOC is invoked and a structure similar to that in Figure 4 is generated. This structure is then passed back through the definition of locate and in this case returned hack to be associated with the surface proposition from which it was invoked. Thus, the structure generated by *L0C becomes associated with the use of the verb locate. The term is when is an action of SOL which carries out the details of passing back the newly constructed structures.

Change-of-States. The next simplest type of verb component is that of the change of state where no particular causative component is specified or implied. For example:

The train moved out of the station at 3 o'clock. C2)

In this sentence the subject, train, is the object of moved, not the causative agent. Letting "CHANGE he the underlying primitive indicating change of state, we illustrate the network structure for sentence (2) in Figure 5.

Figure S.

We want to define *CHANGE in such a way that it constructs structures like those shown in Figure 5. The features of these structures are: 1) indicate that the former state (FROM-STATE) terminated at the time of the change; *2)* indicate that the final state (TO-STATE) was initiated at the time of the change; 3) construct and return with a new token node for change with each of the arguments filled with the appropriate structures. The SOL definition of "CHANGE is this:

```
Define *CHANGE as operator.
*CHANGE FROM-STATE S1 TO-STATE S2 AT-TIME T.
Understand that S1 ended at T.
Understand that S2 started at T.
Return with newtoken for "*CHANGE" "FROM-
   STATE" SI "TO-STATE" S2 "AT-TIME" T.
```

We are now ready to define the intransitive (i.e., non-causative) sense of the verb move. We call this

sense M0VE1 to distinguish it from the general sense of move which contains a causative component. The non-causative sense simply indicates a change from one locative state to another one. The SOL definition for M0VE1 is this:

```
Define as predicate M0VE1.
X M0VE1 (FROM-LOC L1 TO-LOC L2 AT-TIME T).
Iswhen a "CHANGE from the state that X
    is located at L1 to the state that
    X is located at L2 occurs at T.
```

Note that when this definition is evaluated, it invokes *LOC twice (through the two uses of locate) and passes the structures built by *L0C to *CHANGE where the final structure of the form in Figure 5 is put together and then associated with the current invocation of MOVE.

Causatives. The prototypical causal verb is, of course, the verb cause itself. The complexity of the causal component of verbs stems from the fact that there are at least three qualitatively different sorts of causes of events. As an illustration, consider the following five sentences:

The cowboy caused Ambrose to wake by putting water on him. (3a)

The cowboy caused Ambrose to wake with a bucket of water. (3b)

The cowboy caused Ambrose to wake. (3c)

The water caused Ambrose to wake. (3d)

Ambrose was awakened by water being put on him. (3e)

Sentence (3a) illustrates the specification of all three types of causes: 1) the agentive cause (the cowboy); 2) the instrumental cause (the water); 3) the method (the putting of the water). Sentences (3b)-(3e) illustrate some of the surface forms in which these causes can appear. We hold the basic underlying model of causatives to be that "someone does something with some instrument." If the event is fully specified, then that event is taken to be the cause; otherwise a dummy act, *D0, is inserted into the structure. Figure 6A-E gives the network representations for the sentences (3a)-(3e).

Figure 6A-E

Note in 6A that the structure for put (from Figure 1) is the event causing Ambrose to wake. When the event is not known it *is* replaced by **D0 with* the agent or instrument properly filled in.

We are now in a position to define cause in such a way that the proper causative structure will be generated whenever the definition of cause is executed:

```
Define as predicate CAUSE.
AGENT X CAUSE PROPOSITION Y (METHOD M
    INSTRUMENT I AT-TIME T).
If M is specified,
    understand that M started at T,
    evaluate M,
    call M "ACT",
else
    call(newtoken for "D0" "AGENT" X
        "INSTRUMENT" I) ACT.
Understand that Y started at T.
Evaluate Y.
Return with a newtoken for "*CAUSE"
    "EVENT" ACT "RESULT" Y.
```

In this definition we first check to see whether the method is specified; if so, we say that it was initiated at the time of the cause, compute the structure associated with the method (by evaluating the procedure MJ, and save that structure in a variable called ACT. In case the method is unspecified, we build a dummy action and store it *in* ACT. We then compute the structure for Y, the caused event (by evaluating the procedure for Y). Using the predicate for the primitive sense of cause, we now link the causative event to the resultant event. Finally, the procedure returns with a structure that represents the entire definition.

Now that we have defined the primitives for the three basic types of components, we can use these as building blocks to define ever broader classes of verbs with increasingly natural definitions. We can, for example, define the verb MOVE as it appears on the surface. The SOL definition of MOVE is this:

```
Define as predicate MOVE.
(AGENT X) MOVES Y (FROM-LOC L1 TO-LOC L2
    METHOD M AT-TIME T).
If X is not specified,
    iswhen Y move! from L1 to L2 at T,
else
    iswhen X caused Y to movel from L1
        to L2 by M at T.
```

Here move is defined only in terms of the intransitive move (MOVE1) and CAUSE. Similarly, we can define the verb put in terms of MOVE so that the structure illustrated in Figure 1 is produced:

```
Define as predicate PUT.
ASENT X PUTS Y AT-LOC L (AT-TIME T).
Iswhen X moves Y to L at T.
```

Note that these definitions do more than simply rewrite one verb in terms of another. The important point about the entire memory model is the type of representational structure that is constructed with the network. With these verb definitions, the primitives build new structures and modify old information. Thus, in the definition of MOVE, the last line performs the processes for CAUSE and also the processes defined for M0VE1. CAUSE both builds a structure for the causal factors and also performs whatever processes are represented by M, the method. The process for M is passed as an argument down from the original sentence that was entered through the parser, through the definitional structure for MOVE, and finally to the definitional structure for CAUSE. There it is finally executed, building whatever network structure the method M represents.

The Three Drugstores Problem

In this section we give an example of one problem being analyzed by our research group. A major feature of the way that a person views the events of the world is in terms of their causal factors. That is, we tend to disbelieve that an event could simply happen by itself; rather, we tend to believe that an event must have a cause. The tendency to give causal reasons for events is important because it affects the ways in which people make use of information. To illustrate the point, we analyze the three drugstores problem.

The basic problem before us was eloquently posed by Abelson and Reich. We paraphrase their version of the problem in this way:

Suppose an individual says a sentence such as,

"I went to three drugstores." (4)

A response based on syntax only might be,

"How did you go to three drugstores?" (5)

A response based on some semantics might be,

"What useful things did you buy in three drugstores?" (6)

But the most natural response ought to be,

"How come the first two drugstores didn't have what you wanted?" (7)

Solving the Drugstore Problem. Just what must the required processes look like *to* be able *to solve* the drugstore problem? To solve the first few levels all that is needed is a pattern-match program that examines the structure of the verb of the sentence and compares the allowable arguments with those actually presented. Thus, in the sentence, "I went to a drugstore," we see that the to-location is provided but not the from-location, the method, or the time. Thus, it is really a simple matter to construct questions like (5).

To be more intelligent a basic decision must be made: Should the missing information be requested? The answer is usually no. In normal conversation information is omitted either because it is assumed to be provided by the preceding or following context or because it is unimportant to the conversation. The pattern-match routines (inside a procedure called comprehend) fill in information by examining the structure of preceding sentences. Sometimes the information in prior sentences might be appropriate to later ones, and sometimes the information given in the present sentence might fill in missing arguments from previous sentences. When missing arguments are noticed, an attempt is made to answer the implicit question provided by their absence through an examination of the data base. In addition, the present input is examined to see whether it can fill arguments missing in the data base being constructed from the conversation.

So far, we have simply investigated a simple means for filling out the syntactic pattern for verbs, albeit with some sophistication in determining when to ask for *mors* information. The next step is more complex. Suppose we wish to determine why someone has gone to the drugstore. Again, we should not simply have to ask why, but rather determine the general reasons for going to the stores. For this point the comprehend routine must be intelligent enough to examine a more general data base. Now a fair amount of inference is required: we need *to* match the basic paradigm with the specific information given by the parsed sentence. *This* is not easy when one considers that many different paradigms will probably be stored. If the sentence had been, "John went to a shoestore," then the same analysis should clearly not yield the query, "What did John buy at the shoestore?" The comprehend routine must be flexible enough to solve this part of the problem by itself. A large amount of world knowledge is needed to solve the general problem.

This brief analysis shows that in order to have intelligent conversation it is necessary to be able to generate internal questions and their answers, Whenever information is missing some attempt must be made to fill in the gap, sometimes by asking appropriate questions, but usually by internal problem solving. In general, information should not be requested by means of a question unless there is some actual need for it at the moment. Moreover, it would appear that the information should be asked from the very highest level down. Thus, the first question asked should refer to the motive and results of the operations being described. Only later should specific details of the method be asked.

In the implementation of the memory model system at the time of this writing, all the levels of analysis can not yet be performed. Basically, the implementation is complete up to the level of the sophisticated internal answering of questions. Thus, it has been an easy matter to implement a question answering routine to ask questions like the following for the input sentence: How did John go to the drugstore? What did he do afterwards? With whom did he go? At the moment, the basic routines to ask such questions as "What did he buy at the drugstore?" are close to operation, but the construction of the system that can ask the question originally posed, "How come the first two drugstores didn't have what you wanted?" still remains some distance away.

The memory representation provides a rich environment for simulating human cognitive processes. The major ideas have been implemented, yielding an active network representation with an English parser that allows interaction with the network and ready extendability. Actual simulations of human cognitive tasks have just begun, and although work is in progress in a variety of areas, no large system has yet been completed. However, for a description of the use of this system in human problem) solving, see the paper by Eisenstadt and Kareev.

Notes and References

1. This research was initially supported by Grant NS 07454 from the National Institutes of Health. Continuing work is supported by the National Science Foundation Grant GB 32235X. We would like to thank Dan Bobrow for his critical comments on an earlier draft of this paper. Requests for reprints or more information may be addressed to either of the authors at: Department of Psychology; University of California, San Diego; La Jolla, California, 92037.

2. The system programming was performed by Robert Olds and Robert Schudy. The parser and the means by which it interacts with the data base and interpreter were developed by David Rumelhart and originally programmed by Chris Murano. Dennis Bell has assisted in later developments along with Olds and Schudy.
3. Rumelhart, D.E., Lindsay, P.H. & Norman, D.A. A process model for long-term memory. In E. Tulving and W. Donaldson (Eds.), Organization of Memory. New York: Academic Press, 1972.
4. Norman, D.A. Memory, knowledge, and the answering of questions. In R.L. Solso (Ed.), Contemporary Issues in Cognitive Psychology: The Loyola Symposium. Washington, DC: Winston, 1973. (Distributed by Wiley)
5. Kaplan, R.M. Augmented transition networks as psychological models of sentence comprehension. Artificial Intelligence, 1972, 3, 77-100.
6. Woods, W.A. Transition network grammars for natural language analysis. Communications of the ACM, 1970, 13, 10, 591-631.
7. Woods, W.A. & Kaplan, R.M. The Lunar Sciences Natural Language Information System. Cambridge, MA: Bolt Beranek and Newman, Inc., Report No. 2265, 1971.
8. Fillmore, C.J. The case for case. In E. Bach and R.T. Harms (Eds.), Universals in Linguistic Theory. New York: Holt, Rinehart and Winston, 1968.
9. Schank, R.C. Conceptual dependency: A theory of natural language understanding. Cognitive Psychology, 1972, 3, 4, 552-631.
10. Chafe, W.L. Discourse structure and human knowledge. In J.B. Carroll and R.O. Freedle (Eds.), Language Comprehension and the Acquisition of Knowledge. Washington, DC: Winston, 1972. (Distributed by Wiley)
11. The general representational format for verbs suggested in the current paper was developed in a working seminar attended by Adele Abrahamson, Dedre Gentner, Jim Levin, Steve Palmer and David Rumelhart which met weekly from January through July, 1972.
12. Abelson, R.P & Reich, C.M. Implication modules: A method for extracting meaning from input sentences. Proceedings of the International Joint Conference on Artificial Intelligence. Washington, DC, 1969, 641-647.
13. Eisenstadt, M. & Kareev, Y. Towards a model of human game playing. Proceedings of the Third International Joint Conference on Artificial Intelligence. Stanford, CA, 1973.

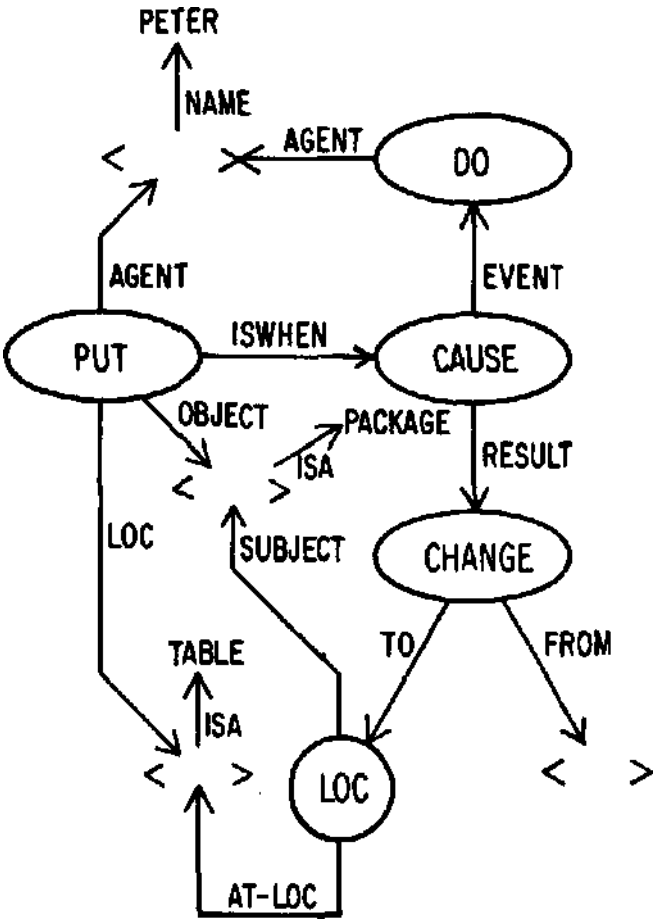
PETER
NAME
AGENT
DO
AGENT
EVENT
PUT
ISWHEN
CAUSE
OBJECT
PACKAGE
ISA
RESULT
LOC
SUBJECT
CHANGE
TABLE
TO
FROM
ISA
LOC
AT-LOC

FIGURE 1

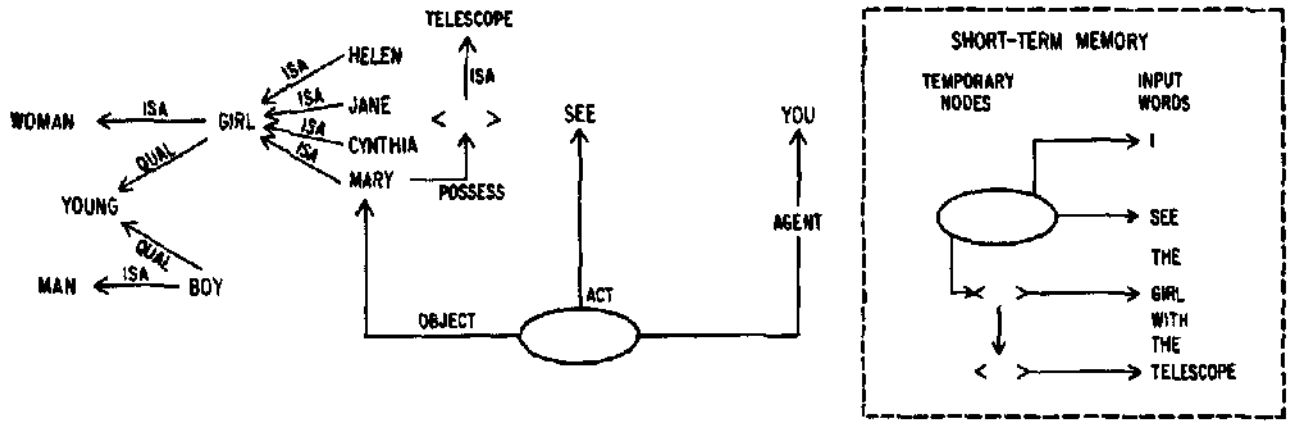

FIGURE 2

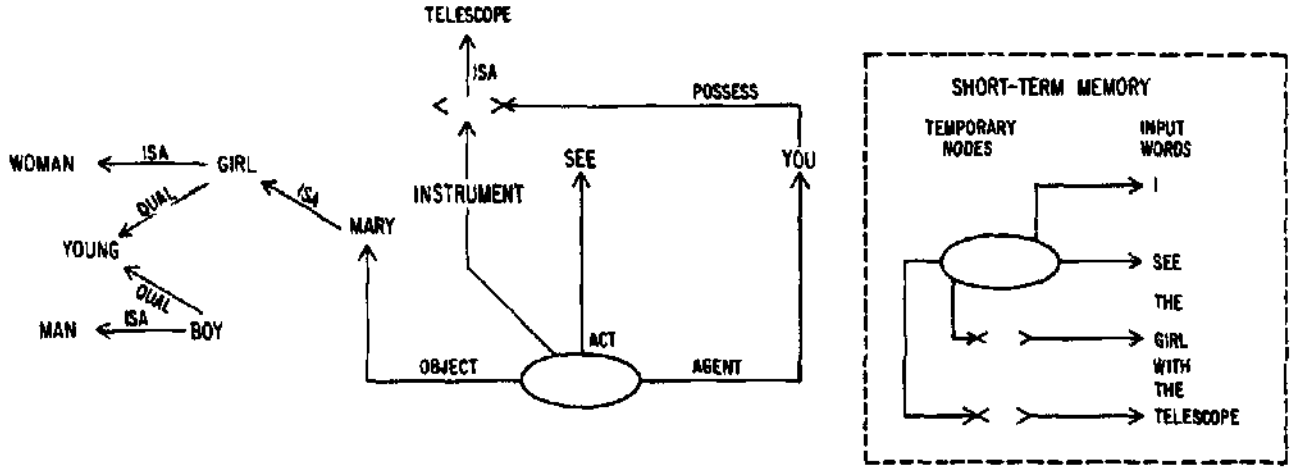

FIGURE 3

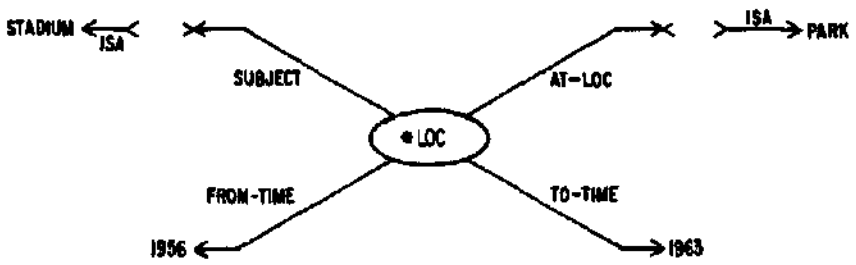

FIGURE 4

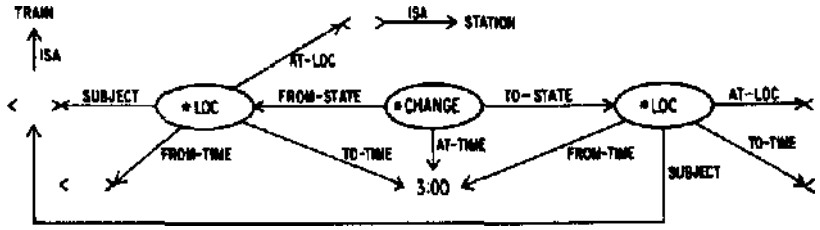

FIGURE 5

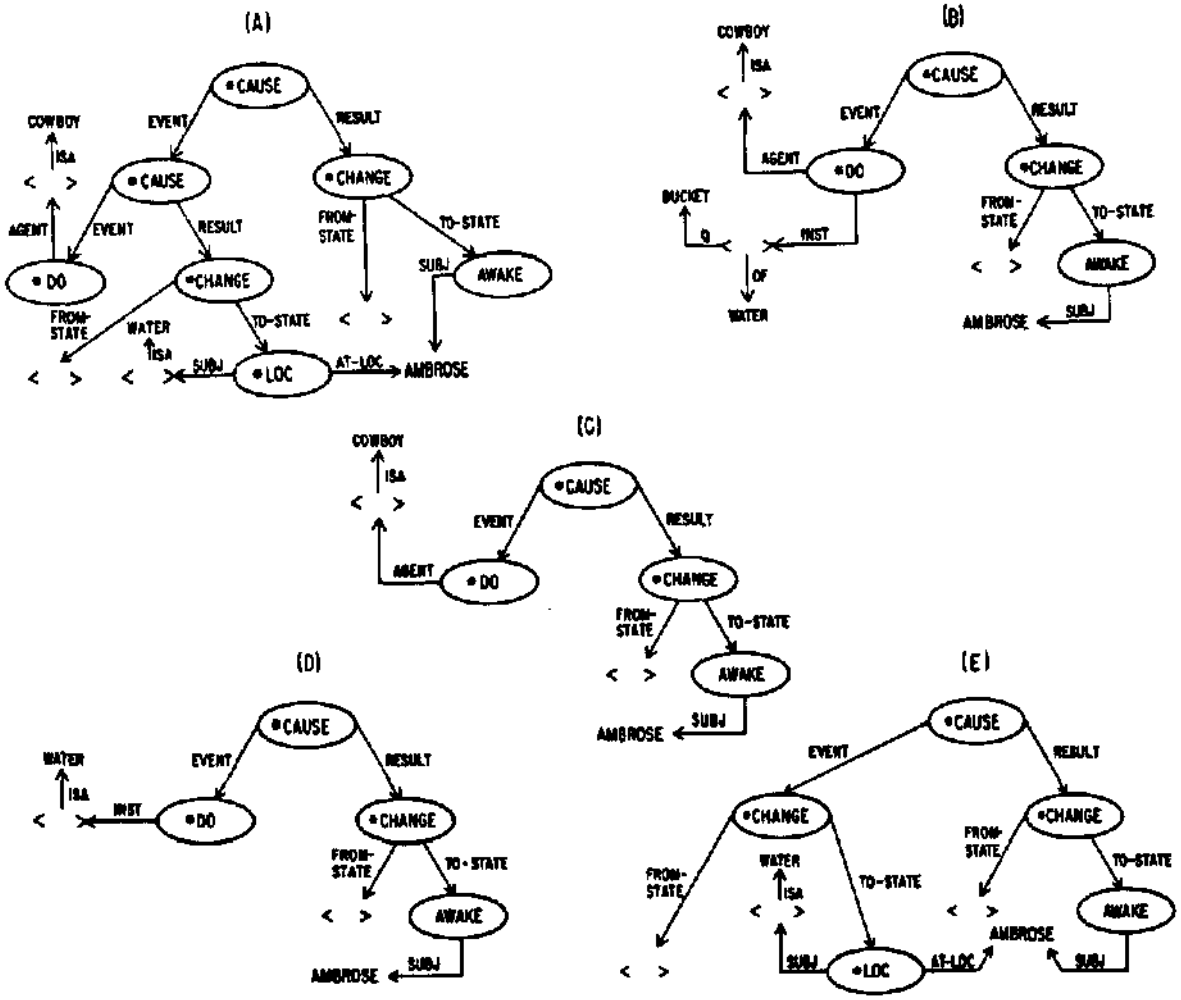

FIGURE 6