MAPPING ENGLISH STRINGS INTO MEANINGS

Robert F. Simmons

Technical Report NL 10

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

The meaning of an English string is taken to be the portion of a dynamic model into which it maps or from which it is generated. A stepwise continuity of linguistic models is shown from a keyword scanner with arbitrary structure building capability to the complexity of an interpreter that can accept a program containing predicates, variables, functions and subroutines to accomplish deep semantic or conceptual analysis.

Keywords: models, meaning, English strings, syntactic analysis, linguistic models, keyword scanners.

ACKNOWLEDGEMENTS

I am grateful to Robert A. Amsler for his unpublished experiments showing the simplicity and power of the keyword approach and to Gary Hendrix and others for helping me to understand set-theoretic modelling techniques.

MAPPING ENGLISH STRINGS INTO MEANINGS*

Meaning - 1a. the thing one intends to convey esp. by language: import

1b. the thing conveyed esp. by language: purport

This is the first sense defining "meaning" in Webster's 7th New Collegiate Dictionary. In terms of our computational understanding of English, the definition is perfectly valid if we simply change the word, "thing" to "model". Two systems that communicate in English are conveying portions of their models of the world relevant to a discourse by mapping the model into English and mapping English strings into a model. The import of an English string is the model from which it is generated; the purport of the string is the model into which it is semantically parsed.

Following Minsky "We use the term "model" in the following sense: To an observer B, an object A* is a model of A to the extent that B can use A* to answer questions that interest him about A." (Minsky 1968).

From a psychological viewpoint such as that taken by Lindsay and Norman (1972), by Quillian (1968) or by Simmons (1972) the meanings of language strings are structures of objects, relations and processes in a model of human memory or cognitive structure. From the artificial intelligence viewpoint models of human cognitive structure are computational representations of mathematical models of what the person knows, and are not essentially different in kind from models of the micro-worlds of robots such as the one developed by Winograd (1972).

---

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

All of these models have a dynamic quality. Although a meaning is a portion of a model it is not to be construed as a static data structure; it is, instead, the operation of certain processes that are modelled. For example a simple declarative sentence requires that its words be identified and mapped into objects in the world-model. If it adds new information, new relational connections must be constructed between objects. Additional processes such as checking for contradictions, generating implications, etc. may also be part of the meaning. For an imperative statement the model into which it is mapped may be operated as a program that can affect both the model and the world that is modelled. The model representing the meaning of a question is a program that actively seeks a certain relational path through the model of the world.

The models that have been most satisfactory for language understanding systems are set theoretic in nature. They are mathematically characterized as a set of objects, a set of relations among these objects, and sets of predicates and operations defined over the objects and relations. Models can be trivially simple or immensely complicated. A simple model with which we will deal at some length includes a robot, R, a room, R1, the rest of space, R2, the relations, INSIDE and OUTSIDE with corresponding predicates INSIDE(R,R1), OUTSIDE(R,R1), and the operation GO(R,R1). Such a model can be used to represent the position of robot, R, inside or outside of R1, can change that position and can determine with the use of predicates the present location of R.

Vastly more complex models of the world may still have this same logical structure of objects, relations and operations but the numbers

of objects and relations may become very large, the relations may become many-valued and the operations may become heuristic, goal-seeking, advice-taking, etc. and even become unpredictable in their interaction. Such complex models are typical of those constructed in PLANNER, QA4, CODER, CONNIVER, etc. (See Baltzer 1972 for a review in this area.)

Our tiny one room, one robot model can easily represent the meaning of such natural language statements as the following:

Where is the robot?

Is the robot in the room?

Go inside the room.

Go outside the room.

In order to understand such strings, the micro-world model must be augmented by a model of the linguistic structures used in the English subset.

The linguistic model also can range from the very simple to the very complex. Perhaps the simplest model takes literal words as its objects, sequences of words as relations and predicates, and transformations into operations on the world-model as its operations. Such a linguistic model can be written as follows for the four statements above.

(SEQUENCE Where, robot) → (Get R,INSIDE)

(SEQUENCE Is, robot, in, room) → (((EQ (GET R INSIDE)R1)→ "YES)
(T →"NO))

(SEQUENCE Go, Inside, room) → (GO R R1)

(SEQUENCE Go, Outside, room) → (GO R R2)

This linguistic model assumes a LISP interpreter, the use of a property list structure to show the relations that hold between pairs of objects, the functions PUT and GET which operate on the property list

to form and retrieve relations, and a function, SEQUENCE, which determines whether the ordering of its arguments is true for the input sentence. SEQUENCE is not concerned if other elements intervene between its arguments. The right-pointing arrow indicates a conditional such that if the left part is satisfied the LISP program on the right is operated. In this linguistic model, words are LISP atoms, the sequential relations are explicitly shown in the left half of the rules, SEQUENCE acts like a predicate on the ordering of words in the input string, and the right half of the rule is the transformation into a meaningful operation on the world model.

The world model is a few programmed operations and a list of objects and relations represented on the property list.

Such keyword based linguistic models have found wide utility in question answering research starting with Bobrow (1968), in natural language conversational systems such as ELIZA (Weizenbaum 1966) and PARRY (Colby 1971), and are basic to most Computer Aided Instruction programs. They are very useful for dealing with small subsets of natural language where only a few meanings need be extracted. They require little in programming complexity, but in their simplest form they must have a rule for every varying sequence of significant words. For example if the model above is to understand the following additional sentences,

Go into the room.

Go to the room.

Go outside the room.

either three more rules must be written or additional programs must be added to allow more general statements of the rules.

This paper is primarily concerned to outline a step-by-step progression of program requirements that will show a conceptual continuity from keyword parsers with simple models to deep structure analyzers with complex goal-seeking medels. There will be little attempt to treat the world-models in full detail as is done by Winograd or Balzer; instead emphasis will be placed on the varying forms and structures of the linguistic models.

II. Escalating the Keyword Model

In this section the linguistic model is generalized by the introduction of predicates, variables, subroutines and data structures to show how its power can be increased to that of a full programming language with the capability of deriving deep semantic or conceptual structures as the meanings of language strings.

Synonymous Forms. In unrestricted English the problem of defining when two or more forms are synonymous is a very difficult one that depends on linguistic and environmental contexts. In computational subsets, however, two or more forms can be defined as synonymous it it is desired that they have the same effect and if that definition does not introduce confusions with other meaningful statements in the formal subset. Thus in the elementary model introduced earlier, we would probably wish to consider the following strings to be synonymous.

Go to the room

Go into the room

Go to R1

Go into R1

Go inside R1

Go in the room

By substituting "the room" for R1 and vice versa in each example, a total of eight pragmatically equivalent paraphrases results. The simplest linguistic model described earlier would require eight rules to recognize these variations although they all map onto the same meaning.

Introduction to the rule system of the Boolian predicate, OR, allows us to write a single rule for all the variations as follows:

(SEQUENCE Go, (OR to, into, inside, in), (OR R1, room)) → (GO R R1)

This approach does not require the introduction of a lexicon, but although it reduces the number of rules to be written, each element of a synonym group must be explicitly listed in the predicate.

A slight change in the form and interpretation of the rules provides the use of variables for additional generalization. We might desire, for example, to write one rule covering the following statements:

Go outside the room

Go inside the room

Go outside

Go inside

Go to the room

Go to R1

Go to R2

By using variables as references to elements in the left part of the rule, the right half can generalize from the representation of a single meaning to representing a whole family of meanings. One rule covering the set of statements above is as follows:

(SEQUENCE Go, (OR inside, outside, to), (OR R1, R2, room))

→ (Go R, 2nd, 3rd)

The terms "2nd" and "3rd" are variables that refer to the position of elements in the left half of the rule. For the operation GO (now programmed for 3 arguments) the values of these variables will be the actual terms that occurred in the left-half positions that are referenced by "2nd" and "3rd". Thus the call to GO is derived with different argument values as follows for the several sentences above:

(GO R, outside, room)

(GO R, inside, room)

(GO R, outside, NIL)

(GO R, inside, NIL)

(GO R, to, room)

(GO R, to, R1)

(GO R, to, R2)

The operation, GO, can be programmed to accomplish the desired changes in relations among objects for each sequence of arguments.

The use of variables is accomplished quite easily since it is a central feature of all programming languages. In LISP, a rule interpreter called RI is written so that it binds the arguments of SEQUENCE to a fixed list of variable names; 1st, 2nd, 3rd,...nth, where n is some definite number that limits the total number of arguments that can be dealt with. As RI interprets the right half of the rule, references to 1st, 2nd, etc. are in fact references to the bindings of those variables.

A consequence of introducing Boolian predicates and variables is that a single rule can map a large set of language strings into a large set of meanings -- i.e. operations with particular sets of arguments. This power is not wholly desirable in that if used to minimize the number of linguistic rules, the complexity of the logic programmed into the

the operations must increase to include conditions identifying the linguistic forms of arguments. In the example above, the operation, GO, must recognize that "to R1", "to, room", "inside room", "inside NIL" etc. result in the same changes in the world-model.

For most purposes it is obviously undesirable to include this much linguistic information in the definitions of operations on the model. One solution is to transform synonymous linguistic forms that occur in the left half of the rule into a canonical form recognized in the definition of the operations. Thus "room" or R1 would always transform to R1, and "in", "into", or "inside" can all have the canonical form IN1 for the particular model. This effect can be accomplished without introducing a dictionary by adding a transformational logic to the interpreter, RI. The form of the rule changes slightly as a consequence to allow for the pairing of synonymous forms with a canonical form in a manner such as the following:

(SEQUENCE (Go GO) (OR((in,inside,into)IN1)),

(OR((room,R1) R1))) → (1st, 2nd, 3rd)

The operation GO has now been redefined to drop the argument R and to accept the argument IN1 or OUT1. For dealing with synonymies this rule using variables adds nothing but unnecessary complexity. The earlier form without variables can be used to map the synonymous strings onto, (GO IN1 R1) with less programmed machinery.

The form with variables however, allows the writing of the following complex rule to deal with most of the variations in strings that map into GO and its various arguments:

(SEQUENCE (Go GO), (OR((in, into, inside) IN1)((out, outside, from) OUT1)), (OR((room,R1)R1)(R2 R2)))

→ (1st, 2nd, 3rd)

But attempting to account in one rule for all the variations in the arguments for an operation makes each rule long and quite difficult to write. We can either restrict ourselves to the relatively easy form of rules without variables, or introduce a lexicon that can serve as the basis for substituting canonical for linguistic forms. It will be seen that adding a lexicon greatly increases the power of the system and integrates the linguistic model with the model of the world.

A Simple Lexicon. For some purposes a very elementary form of lexicon can be used to map each canonical form used in an operation into elements from a set of words as follows:

IN1 - CF - (in inside into)

OUT1 - CF - (out outside from)

R1 - CF - (room R1

R2 - CF - R2

GO - CF - (go proceed travel)

The rule form now appears as follows:

(SEQUENCE GO, (OR (IN1, OUT1)), (OR, R1, R2)) → (1st,2nd,3rd)

This rule and the example lexicon can translate such statements as the following:

Go into the room → (GO IN1 R1)

Go from the room → (GO OUT1 R1)

Go inside R2 → (GO IN1 R2)

Proceed outside R1 → (GO OUT1 R1)

etc.

In order to use a lexicon the interpreter RI, must have been augmented with a logic that looks up each element of its rule in the lexicon, and tests the current word in the input sentence as a member of the CF set associated with that canonical form. The canonical form can now be seen to be an object in the world model and its CF set contains words which are objects in the linguistic part of the model.

The canonical form can also be seen to be a class name for a set of words. The lexicon can be generalized so that any word may belong to various classes -- such as part of speech and feature-value classes such as gender, number etc. If we change the orientation of the lexicon so that each English word is an entry, then rules for recognizing sequences of words or idioms to map them onto canonical forms may be included as Woods shows in his lexical structure, (Woods 1972); or verbs may have sequence rules associated with them as in Thompson (1973), to identify a particular set of arguments.

In short, the addition of a lexicon, usually in the form of an attribute value structure, lifts words to the full status of objects in the model and numerous relations between them and other objects may be encoded to greatly increase the complexity of the language strings that can be understood.

A More Complex Model. To carry our generalization of keyword grammars its last few steps it is convenient to set up a more complex world-model to make worthwhile the additional linguistic machinery that is introduced. The model is expanded as shown in Figure 1 to a world of three rooms, objects R1, R2 and R3. Three boxes, B1, B2, and B3 are added. The robot, R, and the boxes may be in the relation INSIDE* to a room; the rooms may have the relation CONTAIN* to the robot and the

B1	NAME	BOX
	COLOR*	RED
	INSIDE*	R1
	ONTOP*	NIL
	UNDER*	NIL
B2	NAME	BOX
	COLOR*	BLUE
	INSIDE*	R1
	ONTOP*	NIL
	UNDER*	NIL
B3	NAME	BOX
	COLOR*	GREEN
	INSIDE*	R1
	ONTOP*	NIL
	UNDER*	NIL

R1	NAME	ROOM
	COLOR*	GREEN
	CONTAIN*	(B1 B2 B3)
	DOORWAY*	R2
R2	NAME	ROOM
	COLOR*	BLUE
	CONTAIN*	NIL
	DOORWAY*	(R1 R3)
R3	NAME	ROOM
	COLOR*	RED
	CONTAIN*	NIL
	DOORWAY*	R2
R	NAME	ROBOT
	INSIDE*	R3

```
PUSH*(OBJ,REL,GOAL)
STACK*(OBJ,REL,OBJ,PLACE)
UNSTACK*(OBJ,REL,OBJ,PLACE)
GO* (PLACE
     OBJ  )
```

Figure 1. Objects, Relations and Operations of a Micro-world Model

boxes. Boxes may be in the relation ONTOP* and UNDER* to each other. Rooms have a relation DOORWAY* to other rooms. The robot participates in the operations PUSH*, STACK*, UNSTACK*, and GO*. The rooms and the boxes each have a relation COLOR* to the verbal objects red, green or blue.

Although it is not particularly relevant to our linguistic considerations, the operations are goal-oriented. For example if the robot is in R1, the blue box is on top of the green one, and they are both in R3; when the robot is told to push the blue box into R2, PUSH* is programmed so that it will call GO* to take the robot into R3, unstack the boxes, then push the blue box into R2, possibly going back through R1 if the doorways require it. A method of programming goal-oriented functions in MICROPLANNER that is equally suitable for LISP is clearly described by Winograd.

An immediate consequence of this more complicated world model is that a single word may now stand for several similar objects -- e.g. B1, B2, and B3 are all named "box" and R1, R2 and R3 are all named "room". The concepts of number -- one, two or three boxes -- and of distinguishing features -- red box, green room -- are introduced. We may also use indirect reference as in "Stack the box in the blue room on the one in the red room". The linguistic model must have enough power to translate the above sentence into

(STACK* Bi ONTOP* Bj Rj)

First a simpler example: "Put the red box in the blue room". We can use the interpreter RI that we have so far developed if we augment it with a function, SELECT, that takes a determiner, a color and an

object name as arguments and returns the set of objects in the world model that they designate. A rule to translate the example sentence into the operation indicated is as follows:

(SEQUENCE (AND OP (D PUSH*)), DET, COLOR, OBJECT, REL, COLOR, PLACE)

→ (1st, (SELECT 2nd, 3rd, 4th), 5th(SELECT 6th, 7th))

The arguments of the left half of the rule are word classes and predicates. The first argument, AND(OP,(D PUSH*)) requires that the first word, "Put", be an operator that has the denotation PUSH*. Figure 2 shows the necessity for this predicate in that the lexical entry for "put" has two DENOTE values, PUSH* and STACK*. The right half of the rule contains the variables 1st, 2nd, etc. referring to arguments of the left half. The interpreter RI has the task of taking each word class name as an implicit predicate on the word in the string and binding to 1st, 2nd, etc. the value of DENOTE in its lexical entry. Other predicates such as (D PUSH*) can be programmed as functions that return NIL or a value -- such as PUSH* -- to be bound by RI to the current variable.

By consulting Figure 2 we can see that RI will bind the DENOTE values for the arguments in the left half of the rule to result in the following right half:

(PUSH* (SELECT DEF*, RED, (B1 B2 B3)), INSIDE*,

(SELECT DEF*, BLUE, (R1 R2 R3)))

When SELECT evaluates its arguments, the right half appears as follows:

(PUSH* B1, INSIDE*, R2)

These are acceptable arguments for the operator PUSH* and the result of its operation is to change the model of Figure 1 in the following respects:

```
BOX     CLASS    OBJECT                    RED     CLASS    COLOR*
        DENOTE   (B1 B2 B3)                        DENOTE   RED

ROOM    CLASS    PLACE                     GREEN   CLASS    COLOR*
        DENOTE   (R1 R2 R3)                        DENOTE   GREEN

PUT     CLASS    OP                        BLUE    CLASS    COLOR*
        DENOTE   (PUSH* STACK*)                    DENOTE   BLUE

STACK   CLASS    OP                        THE     CLASS    DETERMINER
        DENOTE   STACK*                            NBR      S/PL
                                                   DENOTE   DET*

IN      CLASS    REL                       ONE     CLASS    DET
        DENOTE   INSIDE*                           NBR      1
                                                   DENOTE   DET*

INSIDE  CLASS    REL                       TWO     CLASS    DET
        DENOTE   INSIDE*                           NBR      2
                                                   DENOTE   DET*

ON      CLASS    REL                       COLOR   CLASS    OBJECT
        DENOTE   ONTOP*                            DENOTE   (RED GREEN BLUE)
        IDIOM    (SEQUENCE ((ON TOP) ONTOP*)
                           ((ON TOP OF) ONTOP*)
                           ((ON BOTTOM) UNDER*)
                           ((ON BOTTOM OF) UNDER*))
```

Figure 2. Portion of Lexicon for a Linguistic Model

B1 INSIDE* R2

R2 CONTAIN* B1

R1 CONTAIN*(B2 B3)

We have now generalized the original keyword machinery to include Boolian predicates, generalized predicates, (i.e. word classes and features), variables, and functions. The reader will have noticed that about all that is left to make the system a complete programming language is the addition of subroutines and labels for the rules. The nature of this addition can be seen by considering the following example:

"Put the red box in the blue room in the red room." We could simply write a longer rule of the form already described -- one with 12 arguments in its left half. This approach would make rule writing a fairly tedious process since all variations of English strings of word-classes for the subset would have to be explicitly specified. The introduction of subroutines offers an obvious saving. We can at this point introduce the notion of a Noun Phrase or NP rule and refer to it in the following manner:

(P1 (SEQUENCE (AND OP (D PUSH*)), (CALL NP1), REL, (CALL NP2))

→ (1st 2nd, 3rd, 4th))

(NP1 (SEQUENCE DET, COLOR, OBJECT, REL, COLOR, PLACE)

→ (RETURN(SELECT 1st, 2nd, 3rd, 4th, 5th, 6th)))

(NP2 (SEQUENCE DET, COLOR, PLACE) → (RETURN (SELECT 1st 2nd 3rd)))

Notice that SELECT has been generalized to n arguments and that the functions CALL and RETURN have been introduced to deal with subroutines.

At this stage we can skip the detailing of further changes to the

interpreter, RI, since it is now required to be roughly equivalent to such early string processing languages as Yngve's COMIT (1961) and Bobrow's METEOR (1968). It is more to the point of the present discussion to show how it relates to such recent systems as Winograd's PROGRAMMER and Wood's augmented finite state transition parser.

The graph of Figure 3 shows a network representation similar to the last example rule. The circles are states and the arcs are characterized by predicate tests printed above and operations printed below. The left pointing arrow stands for an assignment operation. The function, GETF returns for the current word in the string, the value of the feature which is its argument -- thus, (GETF DENOTE) for "box" returns (B1 B2 B3). The arcs such as (POP(SELECT 1st, 2nd, 3rd)) assign the value of the SELECT function to a register named * and pass it back to a PUSH arc which called a particular subnet such as NP2 as a subroutine.

The essential similarity between what is accomplished in the example rule and an equivalent Woods net is emphasized by the introduction of Woods' functions GETF, PUSH and POP. Winograd has already discussed the programming equivalence of Woods' network representation and his own PROGRAMMER system (1972, p 201). Since both of these systems have been shown effective to handle complex grammars for important subsets of English, we need not attempt to develop our METEOR-like generalization of keyword grammars further. The point has been made that starting with a keyword scanner that has arbitrary structure building capabilities, we can generalize that scanner into an interpreter that can accept a program containing arbitrary predicate conditions and operations on the elements of a string, and can use variables and subroutines. Both Woods

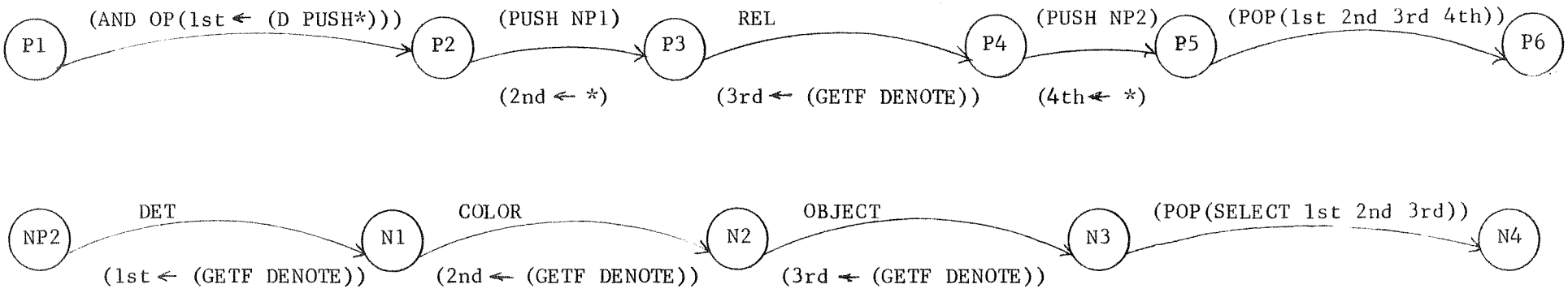

Figure 3. Augmented Finite State Net Representation of Grammar Rules

and Winograd have shown that such a system is powerful enough to construct deep semantic structures to represent the meanings of English sentences.

III. Discussion and Conclusions

A step-wise continuity has been outlined between elementary keyword analyzers and those very sophisticated parsers that provide deep structure analyses of English strings. The essential feature in common is that a keyword string set up as a rule, is predictive in nature. Such a rule, if matched by its first element to a language string, predicts that the rest of the string will be bound by the constraints of the rule. If the constraints on the string fail at any point, another rule is tried. If all constraints are met, transformations may be applied to the string, or any desired program may be operated to construct a representation of the string's meaning. When such rules are generalized to use variables, predicates and subroutines, they can be arranged either as a program in a pattern-operation language such as COMIT, as a LISP program as in PROGRAMMER, or as an augmented network of conditions and operations as in a Woods network.

A few observations can be noted about models. In this paper the view has been taken that underlying the meaning of natural language strings there is a model of some world to be talked about. The models used as examples have been trivialized robot worlds consisting of one or three rooms and a few objects whose inter-relations can be manipulated. Hendrix (1973) shows a generalization of the approach for modelling ordinary textual event descriptions. Some psychologists such as Norman (1972) extend this modelling approach to the description of more complicated worlds such as an ordinary kitchen. What appears significant to me is that

the modelling approach that was outlined is a very ordinary set-theoretic description that appears to have no significant limiting features in its applicability to any domain of discrete objects.* The resulting models are mapped easily into such computational structures as property lists, predicates and programmed operations. One point is that the modelling method and corresponding programs are easily teachable techniques. Constructing actual detailed models of significant domains, in contrast, is apparently a tedious task subject to many errors. Modelling the domain of a single paragraph of text requires deep thought about the meanings that the author expresses in his words and sentences.

The final observation is that corresponding to a model of the world there is a model of a language for talking about it. For English, such models include as objects, words, word classes and features. As relations there are word sequences and set memberships. The operations of linguistic models are generally transformations from a string of words into operations that change the relations that exist among objects denoted by the words in the relevant world-model.

A most important point that derives from this analysis is that simple models of the world require only correspondingly simple models of language. The keyword approach may be quite adequate for talking about a world model whose objects are words, sentences, paragraphs, books, etc. That is, indexing applications and many information retrieval and CAI applications currently fit this limitation and may be adequately treated with minimal linguistic machinery. If a robot were actually

---

* but is not easily applicable to such phenomena as waves passing through fluids or wheels rolling along surfaces.

limited to a few rooms, a few objects and a few operation, little would be required of its language model. But the fact is that any deep description of even a miniscule world gets into such incredibly complicated inter-relations among its elements, that the linguistic model and its interpreter needs the full power of such systems as PROGRAMMER and the Woods network.

REFERENCES

Baltzer, R. Automatic Programming, Information Sciences Institute, Item 1 ARPANET.

Bobrow, D. G. "Natural Language Input for a Problem Solving System" in Minsky, M. (Ed.), Semantic Information Processing. 1968.

Colby, K. M., Weber, S., and Hilf, F. D. "Artificial Paranoia" Artificial Intelligence Vol. 2, #1, Spring 1971.

Hendrix, G. G., "Question Answering Via Canonical Verbs and Semantic Models: A Model of Textual Meaning", The University of Texas, Austin, Technical Report NL 12, January 1973.

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

Minsky, M. (Ed.), Semantic Information Processing, M.I.T. Press, Cambridge Mass. 1968.

Quillian, R. "Semantic Memory", in Minsky, M. (Ed.) Semantic Information Processing, 1968.

Simmons, R.F., "Semantic Networks: Their Computation and Use for Understanding English Sentences" in Schank, R. and Colby, K. (Eds.) Computer Simulation of Cognitive Processes, Prentice-Hall, IN PRESS.

Thompson, C.W., "Question Answering Via Canonical Verbs and Semantic Models: Parsing to Canonical Verb Forms" The University of Texas Technical Report NL 11, January 1973

Weizenbaum, J. "ELIZA" Comm. ACM Vol. 9, #1, January 1966, pp. 36-45.

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

Woods, W. A., Kaplan, R M., Nash-Webber, B., The Lunar Sciences Natural Language Information System: Final Report, Bolt Beranek and Newman Inc. Cambridge, Mass. June 15, 1972, BBN Report #2378.

Yngve, V. H., COMIT Programmers Reference Manual M.I.T. Press, Cambridge, Mass., 1961.