PROBLEMS OF A TEXT-BASED QUESTION ANSWERING SYSTEM

TECHNICAL REPORT NO. NL-7

James Ethan Justiss

August 1972

NATURAL LANGUAGE RESEARCH FOR COMPUTER-ASSISTED INSTRUCTION

Supported by:

THE NATIONAL SCIENCE FOUNDATION
Grant GJ 509 X

Department of Computer Sciences

and

Computer-Assisted Instruction Laboratory

The University of Texas
Austin, Texas 78712

ACKNOWLEDGEMENTS

The author would like to express his gratitude to the many people who helped and encouraged him in this work: To Professor Robert Simmons for supervising the thesis and for the learning process he offered, to Robert Amsler and Jonathon Slocum for their constant aid, to Professor Musser for serving on the committee, and to everyone else who made it a worthwhile experience.

August, 1972

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In recent years an exciting area of computational linguistics has been that of question-answering (QA) systems. These are computer programs that will accept an English question and give a meaningful reply. For a survey of QA systems, see Simmons [7]. Like all computer programs, QA systems suffer from input-output and storage-retrieval interface problems. However, in QA systems the interface problems center around the English language and its lack of a precise definition either syntactically or semantically. Everyone who speaks the language has his own unformalized definition in his head.

Three of the most powerful QA systems have been developed by Woods [14], Winograd [12], and Simmons [10]. Woods' system permits a user to ask a question in English concerning the composition of moon rocks. The system replies with a retrieval from its data base on that subject. Winograd's system is a simulated robot in a tiny simulated world of blocks on a table top. The user can ask the robot questions about its world and can command it to perform tasks. The system accepts this input in English and replies to questions in English. Simmons' system PROTOSYNTHEX III, is the closest to the notion of a text-based question-answerer. It accepts statements (in a subset of English) as input to its data base and accepts questions as queries to the data base.

Although these are reasonably successful systems, they all can be easily "broken" by someone who strays into the ~~holds~~ holes in their natural-language interpretation procedures. This is more than a deficiency of the syntactic grammar, which could be improved merely by putting in every special form of a sentence as a case to be checked. The deficiency involves the problem of interpreting meaning, which is unsolved, except in restricted cases -- by limiting the range of meaning, by limiting the vocabulary, by limiting the syntactic variation, by limiting the subject of discourse, or by limiting the size of the world model. A combination of these techniques is usually necessary.

There has been no clear boundary in existing system descriptions between what is a necessary compromise and what is merely left to be fixed up in a future version of the system. Also many problems of QA systems that have been adequately solved have been presented in such a manner as to leave the reader in doubt as to whether the solution is general or limited. This paper is an attempt to clarify the boundary between limited solutions and general solutions.

A new text-based QA system, called NEWQA, has been designed. Where no solution existed to a design problem, we show our compromises and patches as such. When a solution did exist, it was implemented as well as possible.

The purpose of a text-based QA system is two-fold. First, a text-based QA system takes input from text and organizes

it into a data base format that will support question-answering. Second, a text-based QA system answers questions about the text that is in the data base. This double purpose is of interest theoretically and practically.

On the theoretical side is the problem of how to make a good question-answerer. This problem involves retrieval capability and theorem-proving techniques. On the practical side, the problems are more mundane, but they are not much better understood. A pressing practical problem is the development of a working grammar -- one that will accept a wide range of text. This will be essential some day when a text-based QA system uses a set of books, such as an encyclopedia, as input.

CHAPTER II

PROBLEMS OF QA SYSTEMS

The problems of QA systems fall into three categories. The first one is the problem of interpreting the meaning of natural language input. The second is the problem of accessing what is in the data base. The third is the problem of deduction and inference, which is the theorem-proving problem of Artificial Intelligence. These are interrelated problems, they involve representation of meaning.

In the first category, the interpretation procedures try to create a representation of the meaning of natural-language input. In the process of creating the representation of meaning, the interpretation procedures may have to look in the data base to find concepts that are referenced by the input string. Deductions may also be necessary.

In the second category, the data base procedures look up representations of meaning. Representations of meaning are filed according to similarities of meaning to make the look-up possible. Deductions may be necessary to find the proper index of an entry if it is filed under a different shade of meaning than that assigned to the question.

Finally we arrive at the third category, the deduction procedures. They manipulate representations of meaning to find

other representations of meaning that are also true. Unfortunately, Artificial Intelligence has not yet produced an adequate solution to the theorem-proving problem. No method has been found to meet the requirements so far. We either have to wait for a solution or make do with limited deduction capability.

This is where the ambiguous nature of natural language makes gaps in QA systems. There are many shades of meaning that may or may not be true depending upon what is being talked about and how it is being talked about. Many conditions can affect the truth or falsehood of these representations of meaning. In order to select those that are true we need a deduction capability that far exceeds that of present theorem-proving methods. In QA systems so far, the only deductions that can be made are the ones defined by the builder. Such built-in definitions are a considerable limitation of the ranges of meaning that people can express when they converse.

CHAPTER III

THE OBJECTIVE OF NEWQA

NEWQA was designed to answer questions about a small body of text. It accepts the text as input to build the data base. Then it accepts questions about the material covered in the text. The body of text that it has been designed to handle is the first two paragraphs of STARS, an introductory astronomy book for children.

This objective is a combination of the goals of the systems mentioned in the introduction. The major characteristic of keeping the data input and retrieval functions separate is that the state of the data base is not subject to change while the system is making a reply. Thus NEWQA performs a retrieval function like Woods' system and like PROTOSYNTHEX III, NEWQA creates its own data base from natural language input.

The restriction to input from only a small, specific body of text is helpful to building the parser. In order to use other pieces of text, the parser would have to be augmented to fit the input. This restriction is not a cover-up to avoid an insurmountable theoretical difficulty, but it is rather a convenience. In the parser description we can see that a parser can be made complex enough to handle virtually any syntactic contingency. But in the case of STARS' two paragraphs the text is already so complex that the grammar is fairly incomprehensible, even though the text is an

elementary presentation. Further generality in parsing could increase the size of the grammar by perhaps an order of magnitude. The point would be reached where core space would be filled by grammar programs alone. The result would not be a very balanced system.

The text restriction gives us a chance to keep the system balanced among grammar, lexicon, data base, and utility programs. We are interested in balance because we would like to have a time-shareable system --- where decent response time implies only a few small overlays. (For a description of overlays, see page 9.) The vocabulary of the text amounts to about 300 words (including inflected forms as separate entries). This is about right for the lexical overlay.

Retrieval goals are simple: NEWQA is to reply with as nearly as possible what the original text said --- much as a student would. Thus it cannot answer questions that relate closely to the meaning of the text, but are not closely related to the words in the original sentences. By this means we avoid a massive semantic interconnection problem. It is definitely a patch, but no solution is complete. We have simply said that it is not our goal to achieve that kind of performance.

The objective is to build a system that has undeniable text processing ability. It is to generate its own data base from actual prose --- written for a book and not as a test case. Thus NEWQA attempts to demonstrate that grammar is not an intrinsically

unsolveable problem, but one that is limited by the available core space in a balanced system.

CHAPTER IV

NEWQA'S SYSTEM DESIGN

A block diagram of NEWQA is shown in Figure 1. In U. T. LISP the entire system is composed of four files which operate as three overlays and a randomly accessed storage file. At U. T. a user who desires more than about 32K words of program storage while time-sharing must create overlays. These are user files that contain core images of the LISP system and whatever program the user wants on that overlay. When an overlay program is called from an executing LISP function, the executing function and all of its resident LISP system are swapped out onto temporary storage and the overlay is swapped in. When the overlay program has finished, the overlay returns to secondary storage and execution returns to the calling function.

The first level subsystems of NEWQA are: 1) the Lexicon, 2) the Data Base, and 3) the Parsing Systems. (The parsing system has two sets of grammar, each on its own overlay. There is one for input and one for answering questions, but since they operate similarly, they are discussed as one.) It is a simple design, but as we shall show, for a system to accomplish the goals stated earlier, more complexity often means building more complex patches around unsolved problems. A more complex QA system does not necessarily operate better.

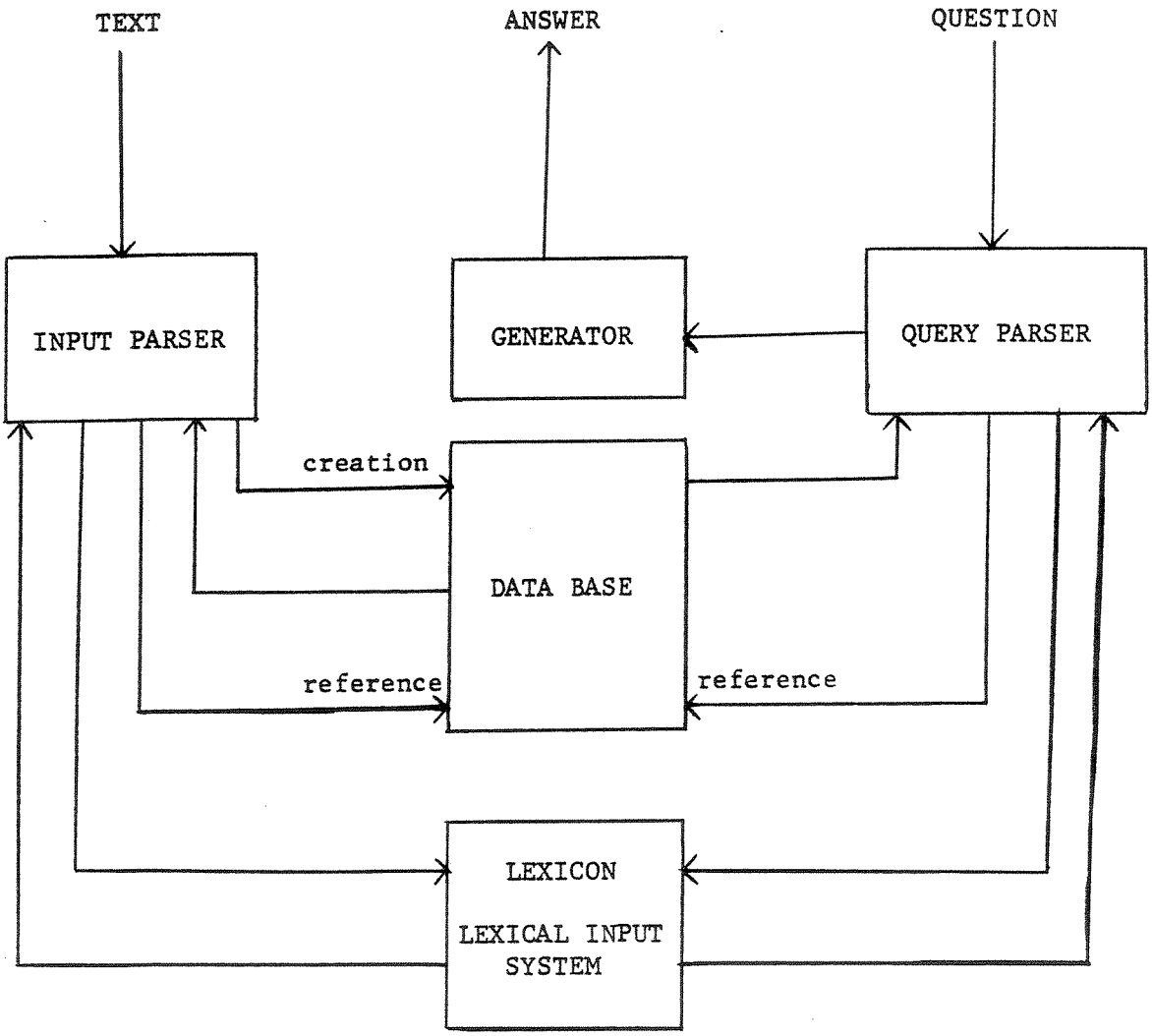

Figure 1

NEWQA system block diagram

An important thing to notice about the block diagram is that there is no data base manipulation subsystem apart from the parser. This is because the parser must refer to the data base as it processes a sentence, so another system is unnecessary.

Storage Structure: Parameterized Semantic Nets

The semantic net structure (Simmons 9 8) has been widely used as an expression of a sentence's deep structure for computational purposes. As a storage structure, a semantic net is composed of a set of triples. In LISP property-list formalism this structure is represented by an atom and a set of properties that consist of an indicator and another atom. Figure 2. shows a semantic net for "Stars and planets have attracted man's attention since earliest times."

There is an indicator and a unique name for every relation in the sentence. This structure is capable of representing the computational structure, but it has an inherent interface difficuly with a program that manipulates it. It requires programs that know what part of the semantic net they are referencing since the indicators differ so much from node to node. Thus the manipulating procedures are inherently part of the interpretation procedures.

By limiting the indicators to a specific few for every node we can reduce the connection between manipulation and interpretation programs. Such a network is called a Parameterized Semantic Net

C1 PI ATTRACTED
MODALITY C2
CA1 C3
THM C4
LOC C5

C2 TENSE PRES
ESS POS
NUM PLUR
FORM PERF
MOOD DECL
VOICE ACTIVE

C3 CONJ AND
ARG1 C6
ARG2 C7

C6 PI STARS
NUM PLUR
DET INDEF

C7 PI PLANETS
NUM PLUR
DET INDEF

C4 PI ATTENTION
NUM SING
DET DEF
POSS C9

C9 PI MAN'S
NUM SING
DET INDEF

C5 PI TIMES
NUM PLUR
DET INDEF
MOD C8
PREP SINCE

C8 PI EARLIEST
DEG SUP

Figure 2

"Stars and planets have attracted man's attention since earliest times." in a semantic net.

C1 PI ATTRACTED

ARGS (C2 C3)

MOD (C4)

FE (VERB PRES PERF)

SM (semantic information)

C2 PI AND

ARGS (C5 C6)

MOD NIL

FE (NOUNPH PLUR COMPOUND

SM (...)

C5 PI STARS

MOD NIL

FE (NOUN PLUR)

SM (...)

C6 PI PLANETS

MOD NIL

FE (NOUN PLUR)

SM (...)

Note:

PI - print image
ARGS - arguments
MOD - modifiers
FE - syntactic features
SM - semantic features

C3 PI ATTENTION

MOD (C7)

FE (NOUN SING DEF)

SM (...)

C7 PI MAN'S

MOD NIL

FE (NOUN FING POSS)

SM (...)

C4 PI SINCE

ARGS (C8)

MOD NIL

FE (PREPPH)

SM (...)

C8 PI TIMES

MOD (C9)

FE (NOUNPH PLUR)

SM (...)

C9 PI EARLIEST

MOD NIL

FE (ADJ SUP)

SM (...)

Figure 3

"Stars and planets have attracted man's attention since earliest times." in a PSN.

(PSN) because the indicators are named for a group of relations. Figure 3. shows "Stars and planets have attracted man's attention since earliest times" in PSN format. Note that in LISP property-lists, PSN's consist of an atom with its properties that are an indicator and (almost always) a list. This format permits them to be manipulated and modified with simpler operations.

Data Base Structure: Entities and Events

The data base structure is identical to PSN structure except that only a certain part of the PSN structure for a sentence becomes a data base entry. Specifically, noun phrases and verb phrases are the retrievable units. A noun phrase is intended to refer to a particular object, even when the object is mentioned in many sentences. Thus the noun phrase must be found in the data base. Retrieval of a verb phrase is necessary to reference an action or an event. These data base entries have been called Entities and Events by Jonathon Slocum*.

Figure 4. shows a noun phrase entity. It has the eight indicators, PI - print image, TOK - token, TPCHN - time and space chain, FE - features (syntactic), SM - semantic features, MOD - modifiers, TMSP - time and space modifiers, and one or more from the set of MOD*, ARG*, or PREP* depending upon what roles the noun phrase plays in the sentence.

---

*Personal Communication.

address	TOK	back chain
	PI	print image
	FE	list of syntactic features
	SM	list of semantic features
	MOD	list of modifiers
	TMSP	time-space modifier
	TPCHN	chain to next occurance
	MOD* or ARG* or PREP*	list of top nodes of entities in which it occurs

Figure 4

A noun entity

address	PI	print image
	MOD	list of modifiers
	FE	list of syntactic features
	SM	list of semantic features
	PREP or CONJ	preposition or conjunction (if any)

Figure 5

A modifier structure

address	PI	print image
	FE	list of syntactic features
	SM	list of semantic features
	MOD	list of modifiers
	TMSP	time-space modifiers
	ARGS	list of argument entities

Figure 6

A verb event

word atom	RT	list of root word (and pointers)
	FE	list of syntactic features
	SM	list of semantic features
	SPEC	special program to be called when PARSE is entered
	SPL	special program to be called when PARSE has completed

Figure 7

A lexical entry

Figure 5. shows a modifier structure.

Figure 6. is a verb phrase event. It has seven indicators; PI, FE, SM, MOD, MOD*, ARG*, PREP*, and TMSP are as before. The verb phrase has no need of either TOK or TPCHN since the verb phrase is not used in successive sentences to refer to the same element of the world as a noun may be. (If is is used in a noun position, a noun entity is created that references it as an event.) It has in addition the indicator, ARGS - arguments, which is an ordered list of where the verb's arguments may be found in the data base.

Lexical Structure

The lexicon is a simple property-list structure with an entry for every word in the vocabulary as shown in Figure 7. Every word has the indicators, RT - root, FE - features (syntactic), SM - semantic features, SPEC - special, and SPL - special. FE and SM are the same for lexical entries as for entities, except they can contain multiple entries for multiple word meanings, i.e. a word can be more than one part of speech and can have more than one meaning within a part of speech. For instance, "pitcher" is a container for water and "pitcher" is also a man who throws a base-ball. RT is a list of one element, the root word, if it is an inflected form. If the word is its own root, RT contains pointers to the inflected forms. SPEC and SPL are places where programs for special words can reside. The word, "and", for instance, has

a SPL program which causes it to parse another unit of the same type as was just parsed. SPEC is for programs which are to be entered (as an interrupt) upon a call to PARSE. SPL is for programs which are to be entered after PARSE has finished its normal operation. This use is identical to Winograd's SPEC and SPECL programs respectively. (See Winograd 12.)

Syntactic Features

Figure 8. lists the syntactic categories into which a word can fall. A word can be in more than one category. As it is parsed into one category or another, the parser selects the features of the word that pertain to that category. Within a category a word can have one or more of the features that are associated with the name of the part of speech in the figure.

There are sixteen parts of speech: NOUN, VERB, ADJ, PREP, ADV, PRON, PROPN, QADJ, CONJ, DET NUM, RELPRON, ORD, PRT, CLASF, and EXPL.

NOUNs can have the features SINGular, PLURal, POSSessive, MASS, and TIME. MASS nouns are non-countable nouns such as "sugar" and "water". TIME nouns have special time meanings, such as "B.C." and "o'clock".

VERBs can also have the features SINGular and PLURal for agreement in number with their arguments. SING3 is the special

```
DEFLIST((
        (NOUN (NOUN SING PLUR POSS MASS TIME))
        (VERB (VERB SING PLUR SING3 FST TRANS INTRANS
               TRANS2 CLOBJ TOOBJ POBJ VBE PRT NOPASS
               AUX QUAX BAUX DAUX HAUX MODAL NEG PRES
               PAST FUT EN ING INF))
        (ADJ  (ADJ COMP SUP))
        (PREP (PREP PREP2))
        (ADV  (ADV ADVMOD VMOD PREPMOD NEG))
        (PRON (PRON THPRON SING PLUR FST SEC POSS NOM
               OBJ DEM POSSDEF NEG REFL))
        (PROPN (PROPN SING PLUR POSS))
        (QADJ (QADJ))
        (CONJ (CONJ SUBCONJ))
        (DET  (DET INDEF DEF QDET DEM QUANT NEG NONUM OFDET))
        (NUM  (NUM SING PLUR NTHAN NAS NAT NUMW))
        (ORD  (ORD))
        (PRT  (PRT))
        (CLASF (CLASF))
        (EXPL (EXPL))
        (RELPRON (RELPRON SING PLUR POSS NOQU))
          ) WDCL)
```

Figure 8

Wordclass categories and their features.

third person singualr form, while FST denotes special first person singular forms as in "be"--"am". TRANS denotes a verb that takes a direct object, such as "hit". TRANS2 indicates indirect object verbs like "give". VBE signals a verb that like "be" or "seem" can take a subject complement. TOOBJ indicates a possible infinitive object on verbs like "fail" and "try", while CLOBJ indicates that a clause object is likely -- often found with "say" and "show". POBJ is found on verbs that require a place designation as in "put the block down" or "put it on the table". ("Put it" is not grammatically well-formed for this system.) PRT indicates that the verb may have a particle as with "look up". NOPASS is a feature of verbs that cannot be passive, such as "cost". AUX is a marker for auxiliaries, which include QAUX (question auxiliaries such as "will" and "did"), DAUX ("do" forms), HAUX ("have" forms), and BAUX ("be" forms). MODALs are "can" and "may". Negation is signaled by NEG on "-n't" verb forms. PRES, PAST, and FUT are the markers for present, past, and future tenses. EN signifies the past participle and ING indicates the present participle. INF is the infinitive form of the verb.

ADJectives can be only ADJ or they may also be COMPar-ative or SUPerlative.

PREPositions may have the feature PREP2 which means that another preposition may follow them. (An adverb phrase may also intervene.) An example of a PREP2 is "as" in "as far as" or "on"

in "on top of".

ADVerbs are distinguished by what they can modify. VMOD signifies verb modifiers such as "-ly" words like "slowly". PREPMOD marks preposition modifiers, and ADVMOD adverbs can modify other adverbs. NEG adverbs are negative ("not").

PRONouns can be SINGular or PLURal in number, and they can be NOMinative, OBJective, or POSSessive in case. FST and SEC denote first and second persons respectively (third person is unmarked). POSSDEF (for possessive definite) pronouns are stand-alone possessive pronouns such as mine, yours, his, ours, etc. REFLexives are the "-self" pronouns such as "myself" and "himself". THPRON marks indefinite pronouns like "something", "anything", and "nothing". "Nothing" is also NEGative. Finally there are the DEMonstrative pronouns, "that", "this", "these", and "those".

PROPN is the abbreviation for proper nouns. They have the features SING, PLUR, and POSS just as nouns do.

QADJ denotes interrogative adjectives and adverbs like "how", "where", and "when".

CONJunctions can have the feature SUBCONJ which means it is a subordinate conjunction. Subordinate conjunctions introduce subordinate clauses within a sentence.

DETerminers include the articles, "a", "an", and "the". "A" and "an" are INDEFinite, while "the" is DEFinite. DEMonstrative determiners are "that", "this", "these", and "those". Question

determiners like "which", "what", and "how many" are marked QDET. QUANtifiers are "any", "every", etc. Quantifiers can also have the feature OFDET, indicating that they can take an "of" phrase. "None" and "no" are NEGative determiners. The feature NONUM says that the word cannot be used with numbers. "None" and "many" are NONUM's.

NUMbers can be PLURal, or if the number is "one" or "1", it is SINGular. All numerical LISP atoms are numbers and all number-naming words like "seven" are numbers. Some other words can also behave like numbers and they are NUM's, too. NTHAN is a feature of number words like "more" and "fewer" which can precede "than". NAS is for number words like "few" which can precede "as" and can be used in "as...as" constructions. NAT is for number words that may follow "at", such as "least" and "most". NUMW is a number word that does not have any of these specific features. "Exactly" is a NUMW.

RELPRON is the name for relative pronouns. They can be SINGular, PLURal, or POSSessive. They can be NOQU if they cannot be used in a question. For example, "which", "who", and "whose" can be used to start questions, but "that" and "whom" are NOQU.

ORDinal numbers like "first" have only the feature ORD.

PRT marks particles like "up" in "pick her up".

CLASF is the name for classifiers -- nouns that can modify other nouns. "Girl" is a classifier in "girl scout".

EXPL marks the expletives "there" and "it" which are

often used to start sentences. Then the true subject of the sentence appears in the object position, as in "There was an old woman who lived in a shoe."

Semantic Features

NEWQA uses a set of semantic features to disambiguate the use of a word in a sentence. The semantic markers are: THING, PHYSOBJ, EVENT, ANIMATE, HUMAN, PLACE, TIME, RELATION, and IDEA. They are stored on the SM indicators of words and entities and used both as selection restrictions and markers.

On a noun the marker words show the meaning of the noun. On modifiers the markers act as selection restrictions to show what the modifier can modify. This provides a selection procedure to decide what modifies what. The SM of a word is intersected with the SM's of its modifiers to give a new SM. If the new SM is empty, the modifier has been assigned to modify the wrong constituent and further analysis is necessary.

For this purpose the simple intersection function is not sufficient to find what all modifiers should modify. Since there are alternate SM meanings for many words, the order of testing words against each other becomes important. For the first meaning may fail to match one constituent and the second meaning will succeed in matching. But the first meaning would succeed if it was matched against another constituent. Therefore the order of matching is:

Match the first semantic meaning everywhere. If it succeeds somewhere, the first place it succeeds is the place it should modify. If it fails everywhere, try the second semantic meaning, and so forth.

As the reader can see, this is not a general solution, but it is a procedure that works simply because the ordering that has been chosen is adequate for the purposes of analysing the STARS text. There is no real solution to this problem in unrestricted text. Without further context, such sentences as: "The man by the painting with the peaches saw me" are necessarily ambiguous to an automatic sentence parser with a non-specific grammar.

Figure 9. shows some sample SM markers for a noun, an adjective, a preposition, and a verb. Unlike other modifiers, prepositions have an SM that limits the meaning that the prepositional phrase can convey. It selects the meaning of its noun-phrase argument. Verbs have a list of possible argument SM markers in each argument position.

Data Base Retrieval

The problem of retrieving a data base entry by everything that it means is unsolved. The ideal would be a data base indexed by every meaning of every entry. But there is no unambiguous and repeatable way to assign every representation of meaning to an entry. If an arbitrary code that describes the possible meanings the word can take is generated for indexing a word or concept, we

```
MAN:
        ((NOUN ((HUMAN) (MAN MEN MAN#S MEN#S MANKIND MANKIND#S))
          ((HUMAN) (MAN MEN MAN#S MEN#S HUMAN HUMANS HUMAN#S
             HUMAN#)))

OLD:
        ((ADJ ((HUMAN ANIMATE THING PHYSOBJ TIME IDEA RELATION)
                (OLD OLDER OLDEST))

IN:
        ((PREP ((PLACE) (IN))
               ((TIME) (IN)))

TRY:
        ((VERB ((( SUBJ HUMAN ANIMATE EVENT IDEA)
                 (OBJ1 THING EVENT IDEA PHYSOBJ))
             (TRY TRIES TRYING TRIED ATTEMPT ATTEMPTED ATTEMPTING
             ATTEMPTS)
         ((SUBJ HUMAN ANIMATE EVENT IDEA))
             (TRY TRIES TRIED TRYING)))
```

Figure 9

Semantic lexical entries