IDIOLECTIC LANGUAGE-ANALYSIS FOR UNDERSTANDING DOCTOR-PATIENT DIALOGUES*

Horace Enea and Kenneth Mark Colby

Department of Computer Science
Stanford University
Stanford, California

ABSTRACT

A programming language is described which is designed to simplify the construction of computer programs to analyze English. This system attempts to merge the best features of pattern matchers and the phrase structure approach to language analysis. Several practival problems which occur in dealing with such a system are described.

INTRODUCTION

Why is it so difficult for machines to understand natural language? Perhaps it is because machines do not simulate sufficiently what humans do when humans process language. Several years of experience with computer science and linguistic approaches have taught us the scope and limitations of syntactic and semantic parsers.(Schank,Tesler and Weber,[8] Simmons,[9] Winograd,[13] Woods[1"]). While extant linguistic parsers perform satisfactorily with carefully edited text sentences or with small dictionaries , they are unable to deal with everyday language behavior characteristic of human conversation. In a rationalistic quest for certainty and attracted by an analogy from the proof theory of logicians in which provability implied computability, computational linguists hoped to develop formalisms for natural language grammars. But the hope has not been realized and perhaps in principle cannot be. (It is difficult to formalize something which can hardly be formulated).

Linguistic parsers use morphographemic analyses, parts-of-speech assignments and dictionaries containing multiple word-senses each possessing semantic features, programs or rules for restricting word combinations. Such parsers perform a detailed analysis of every word, valiantly disambiguating at each step in an attempt to construct a meaningful interpretation. While It may be sophisticated computationally, a conventional parser is quite at a loss to deal with the caprice of ordinary conversation. In everyday *discourse* people speak colloquially and idiomatically using all sorts of pat phrases, slang and cliches. The number of special-case expressions is indefinitely large. Humans are cryptic and elliptic. They lard even their written expressions with meaningless fillers and fragments.They convey their intentions and ideas in idiosyncratic and metaphorical ways, blithely violating rules of 'correct' grammar and syntax. Given these difficulties, how is it that people carry on conversations easily most of the time while machines thus far have found it extremely difficult to continue to make appropriate replies indicating some degree of understanding?

It seems that people 'get the message[1] without always analyzing every single word in the input. They even ignore some of its terms. People make individualistic and idiosyncratic selections from highly redundant and repetitious communications. These personal selective operations, based on idiosyncratic intentions, produce a transformation of the input by destroying and even distorting information. In speed reading, for example, only a small percentage of contentive words on each page need be looked at. These words somehow resonate with the readers relevant conceptual-belief structure whose processes enable him to 'understand[1] not simply the language but all sorts of unmentioned aspects about the situations and events being referred to in the language. Normal written English text is estimated to be 5/6 redundant (Rubenstein and Haberstroh[7]). Spoken conversations in English are prooably better than *507.* redundant(Carroll'). Words can be garbled and listeners nonetheless get the gist or drift of what is being said. They see the "pattern" and thus can supply much of what is missing

To approximate such human achievements we require a new perspective and a practical method which differs from that of current linguistic approaches. This alternate approach should incorporate those aspects of parsers which have been found to work well, e.g., detecting embedded clauses. Also individualistic features characteristic of an idiolect should have dominant emphasis. Parsers represent complex and refined algorithms. While on one hand they subject a sentence to a detailed and sometimes overkilling analysis, on the' other, they are finicky and oversensitive. A conventional parser may simply halt if a

* This research is supported by Grant PHS MH 06645-12 from the National Institute of Mental Health, by (in part) Research Scientist Award (No. 1-K05-K14.433) from the National Institute of Mental Health to the second author and (in part) by the Advanced Research Projects Agency of the Office of the Secretary of Defense(SD-183).

word in the input sentence is not present in its dictionary. It finds ungrammatical expressions such as double prepositions ('Do you want to get out of from the hospital?') quite confusing. Parsers constitute a tight conjunction of tests rather than a loose disjunction. A6 more and more tests are added to the conjunction, the parser behaves like a finer and finer filter which makes it increasingly difficult for an expression to pass through. Parsers do not allow for the exclusions typical of everyday human dialogues.

Finally, it is difficult to keep consistent a dictionary of over 500 multiple-sense words classified by binary semantic features or rules. For example, suppose a noun (Ni) is used by some verbs as a direct object in the semantic sense of a physical object. Then it is noticed that Ni is also used by other verbs in the sense of a location so 'location[1] is added to Ni's list of semantic features. Now Ni suddenly qualifies as a direct object for a lot of other verbs. But some of the resultant combinations make no sense even in an idiolect. If a special feature is then created for Ni, then one loses the power of general classes of semantic features. Adding a single semantic feature can result in the propagation of hidden inconsistencies and unwanted side-effect., as the dictionary grows it becomes increasingly unstable and difficult to control.

Early attempts to develop a pattern-matching approach using special-purpose heuristics have been described by Colby, Watt and Gilbert,[2] Weizenbaum" and Colby and Enea.[3] The limitations of these attempts are well known to workers in artificial intelligence. The man-machine conversations of such programs soon becomes impoverished and boring. Such primitive context-restricted programs often grasp a topic well enough but too often do not understand quite what is being said about the topic, with amusing or disastrous consequences. This shortcoming is a consequence of the limitations of a pattern- matching approach lacking a rich conceptual structure into which the pattern abstracted from the input can be matched for inferencing. These programs also lack a subroutine structure, both pattern directed and specific, desirable for generalizations and further analysis.

The strength of these pattern matching approaches lies in their ability to ignore some of the input. They look for patterns, which means the emphasis of some detail to the exclusion of other detail. Thus they can get something out of nearly every sentence-- sometimes more, sometimes less.

An interesting pattern-matching approach for machine translation has been developed by Wilks.[12] His program constructs a pattern from English text input which is matched against templates in an interlingual data base from which,in turn, French output is generated without using a generative grammar.

In the course of constructing an interactive simulation of paranoia we were faced with the problem of dealing with unedited and unrestricted natural language as it is used in the doctor-patient situation of a psychiatric interview.(Colby, Hilf, Weber, and Kraemer," Colby and Hilf[5]). This domain of discourse admittedly contains many-psychiatricelly stereotyped expressions and is constrained in topics (Newton's laws are rarely discussed). But it is rich enough in verbal behavior to be a challenge to a language understanding algorithm since a variety of human experiences are discussed domain including the interpersonal relation which develops between the interview participants, A look at the contents of a thesaurus reveals that words relating to people and their interrelations make up roughly 707 of English words.

The diagnosis of paranoia is made by psychiatrists relying mainly on the verbal behavior of the interviewed patient. If a paranoid model is to exhibit paranoid behavior in a psychiatric interview, it must be capable of handling dialogues typical of the doctor-patient context. Since the model can communicate only through teletyped messages.the vis-a-vis aspects of the usual psychiatric interview are absent. Therefore the model must be able to deal with unedited typewritten natural language input and to output replies which are indicative of an underlying paranoid thought process during the episode of a psychiatric interview.

In an interview there is always a who saying something to a whom with definite intentions and expectations. There are two situations to be taken into account, the one being talked about and the one the participants are in. Sometimes the latter becomes the former. Participants in dialogues have intentions and dialogue algorithms must take this into account. The doctor's intention is to gather certain kinds of information while the patient's intention is to give information and get help. A job is to be done; it is not small talk. Our working hypothesis is that each participant in the dialogue understands the other by matching selected idiosyncratically- significant patterns in the input against conceptual patterns which contain information about the situation or event being described linguistically. This understanding is communicated reciprocally by linguistic responses judged appropriate to the intentions and expectations of the participants and to the requirements of the situation. In this paper we shall describe only our current input-analyzing processes used to extract a pattern from natural language input. In a later communication we shall describe the inferential processes carried out at the conceptual level once a pattern has been received by memory from the input-analysing processes.

Studies of our 1971 model of paranoia (PARRY) indicated that about thirty percent of the sentences were not understood at all , that is, no concept in the sentence was recosnized. in a somewhat larger number of cases some concepts, but not all, were recognized. In many cases these partially recognized sentences lead to a partial understanding that was sufficient to gather the intention of the speaker end thus lead to output an appropriate response. However, misunderstandings occurred too often. For example:

```
DOCTOR: How old is your mother ?

PARRY:  Twenty-eight
```

PARRY has interpreted the question as referring to his own age and answered by giving his age. The purpose of our new language analysis system is to significantly raise the level of understanding by preventing such misunderstandings while not restricting what can be said to PARRY. We do not expect complete under- standing from this system ~ even native speakers of the language do not completely understand the language.

By 'understanding we mean me system can do some or all of the following:

1) Determine the intention of the interviewer in making a particular utterance.

2) Make common logical deductions that follow from the interviewers utterance

3) Form an idioletic internal representation of the utterance so that questions may be answered, commands carried out, or data added to memory.

4} Determine references for pronouns, and other anaphora.

5) Deduce the tone of the utterance,!.e., hostile, insulting...

6) Classify the input as a question, rejoinder.command,...

The approach we are taking consists of merging the best features of pattern directed systems such as the MAD DOCTOR,[2] ELIZA" and parsing directed systems for example, Winograd,[13] Woods.[14]. By merging the BNF phrase structure approach ot analyzing English with the pattern matching approach, with its attendant emphasis of some concepts to the exclusion of others. The programs to accomplish this are written in ML1SP2, an extensible version of the programming language ML[$P,[5,10] and uses an interpreted version of the pattern matcher designed for a new programming language LISP 70.

The following is a basic description of the pattern matcher. We shall illustrate its operation using examples of problems common to teletyped psychiatric dialogues.

PATTERN MATCHING

Pattern directed computation involves two kind of operations on data structures: decomposition and recomposition. Decomposition breaks down an input stream into components under the direction of a decompostion pattern ("dec"). The inverse operation, recomposition, constructs an output stream under the direction of a recomposition pattern ("rec").

A rewrite rule is of the form;

```
dec --> rec
```

It defines a partial function on streams as follows: if the input stream matches the dec, then the output stream Is generated by the rec. The following *rule* (given as an example only) could be part of a question answering function:

```
How are you ? -» Very well and you ?
```

If the input stream consists of the four tokens:

```
How are you ?
```

the output stream will consist of the five tokens:

```
Very well and you ?
```

REWRITE FUNCTIONS

A rewrite rule defines a partial function, for example, the mapping of some particular token into some other particular token. A broader partial function can be defined as the union of several rewrite rules. A rewrite function definition is of the form:

```
RULES OF <name> -
         decl -* reel,
         dec2 -» rec2,

         decn -» recn;
```

VARIABLES

A function is difficult to define if every case must be enumerated. Therefore, rewrite rules allow variables to appear in patterns. The value of a variable can be either a list or an atom. In this paper the notation:

:X

where X ia any identifier, will denote the variable X. The variables of each rule are distinct from the variables of all other rules, even if their names are the same.

The following definition has only ihree rewrite rules, but handles an unlimrted number of input streams:

```
RULES OF REPLY=
     HOW ARE YOU '? → VERY WELL '?
     HOW IS :X → I HAVEN''T SEEN :X ',
                                  LATELY'.,
     DID :X GO TO :Y '? →
          WHY DON''T YOU ASK :X YOURSELF'?;
```

A variable can appear more than once in a single dec pattern, but it must match identical items at each appearance. Example:

```
RULES OF EQUAL =
        (EQUAL :X :X) → TRUE;
```

ELLIPSIS

To make patterns easier to read and write, the ellipsis symbol ... can be used to stand for an unnamed variable. Thus:

```
IS ... COMING → NO. ... COULD NOT MAKE IT.
```

If an ellipsis (...) occurs several times on a side, it designates a different variable each time. The n'th ellipsis in a dec designates the same variable as the n'th ellipsis in the rec.

Ellipsis is one of the principle ideas of pattern matching. It permits imprecise mathching; that is, the emphasis or ignoring of items.

AUTOMATIC ORDERING OF RULES

The order of rules in a function definition does not specify the order in which the system will attempt to apply them. This ordering operation is handled by a special system ordering function. Consider the rewrite function:

```
RULES OF REPLY =
        I SEE :X → SO WHAT '?,
        I SEE ANN → WOW '!;
```

Both rules would match:

```
I SEE ANN
```

In such cases the more specific rule takes precedence. Thus given:

```
I SEE ANN
```

as the input stream , the output stream would be:

```
WOW !
```

but given:

```
I SEE STARS
```

the output stream would be:

```
SO WHAT ?
```

A literal is more specific than a variable. A variable appearing for the second time is more specific than a variable appearing for the first time in a dec. This is so because the second occurence of the variable must match the same pattern as the first occurence. The precedence function is itself written in rewrites and so is both extendable and changable by the user. Currently precedence is calculated by a left to right application of the above criteria. Therefore, the following function defines the LISP function EQUAL:

```
RULES OF EQUAL =
        (EQUAL :X :X) → T,
        (EQUAL :X :Y) → NIL;
```

SEGMENTS

Sometimes it is desirable for a variable to match an indeterminate number of items. This is notated:

```
::X
```

Use of the double-colon ("::") means that the variable (e.g., X) will match zero or more items. Example:

```
RULES OF APPEND=
        (APPEND (::X)(::Y)) → (::X ::Y);
```

or if the input stream were:

```
(APPEND (A B) (C D E))
```

the output stream would be:

```
(A B C D E)
```

For increased readability the rule could also be written:

```
RULES OF APPEND =
        (APPEND (...) (...)) → (... ...);
```

Another example:

```
RULES OF REPLY =
        WHERE DID ::X GO →
                ::X WENT HOME '.;
```

Therefore,

```
WHERE DID THE CARPENTER GO →
        THE CARPENTER WENT HOME.
```

APPLICATION

One of the main deficiencies of the system in which the MAD DOCTOR was programmed was its lack of adequate subroutining capability. Subroutines may be indicated in the rewrite system as follows:

```
RULES OF LAST =
        () → (),
        (:X) → :X,
        (:X ...) → <LAST (...)>:
```

The "<>" surrounding a pattern means that the current input stream is to be pushed down, that the function indicated by the first token within the brackets is to be entered with the rest of the pattern appended to the front of the input stream, and that the output stream is to be placed into the restored current input stream. Note that MLISP2 functions may be called as well as rewrite functions.

GOALS

To gain the advantage of goal directed pattern matching and computing, as well as the full power of context sensitive grammars, the following form may be used:

```
RULES OF PREPOSITIONAL_PHRASE =
    <PREPOSITION>:P <NOUN_PHRASE>:N
        → (PREP_PH :P :N);
```

The identifer between the angled brackets ("<>") names a rewrite function the rules of which are to be matched against the input stream. When a match occurs the output stream of the goal will be bound to the associated variable. Example;

```
RULES OF PREPOSITIONAL_PHRASE =
        <PREPOSITION>:P <NOUN_PHRASE>:N
                → (PREP_PH :P :N);

RULES OF NOUN_PHRASE =
        TOWN → (NOUN_PH TOWN),
        PALO ALTO → (NOUN_PH PALO_ALTO);

RULES OF PREPOSITON =
        IN → IN,
        ON → ON;
```

and the input stream:

```
IN PALO ALTO
```

the output stream would be:

```
(PREP_PH IN (NOUN_PH PALO_ALTO))
```

OPTIONALS

Many other shorthands exist to simplify writing rules. One useful feature that will be mentioned here is the optional.

```
RULES OF AUXILARY_PHRASE -
        <AUXILARY>iA    [<NEGATIVE>:N3    N1
                (AUX_PH :A [:"N):NI >:
```

If the optional pattern, enclosed in square brackets {"[]"), occurs in the input stream it will be bound to :N. :N1 will be bound to 2. If the <NEGATIVE> does not occur, :N1 will be bound to 1. On the rec side of the rules if :NI is 2 then :N will be placed in the output stream. If it is 1 then nothing is placed in the output stream at that point. Example, given the rule above:

```
DO → (AUX_PH DO)
DO NOT → (AUX_PH DO NOT)
```

MORE EXAMPLES

We have collected a large number of dialogues using our previous program PARRY. These dialogues form a large body of examples of the kind of English which we can expect. Martin Frost, a graduate student in Computer Science, Stanford University, has written a keyword in context program which enables us to isolate examples centered on particular words so that uses of thoses words in context become more apparent. Our general approach is to build a system which can produce desired intreptations from these examples and to incrementally add to the rules *in* the system as new cases are discovered during the running of the program.

Following are some examples of commonly occuring situations and examples of the kind of rules we use to handle them.

QUESTION INTRODUCER

In doctor-patient dialogues it is quite common to introduce a question by the use of a command. The "question introducer" is followed by either a <NOUN_PHRASE> or a <DECLARATIVE_SENTENCE>. For example,

```
COULD YOU TELL ME YOUR NAME?
```

Rather than attempt a literal analysis of this question, which might lead to the interpretation:

```
DO YOU HAVE THE ABILITY TO SPEAK YOUR NAME TO ME?
```

we utilize rules like:

```
RULES OF SENTENCE =
        <QUESTION_INTRODUCER>:Q <NOUN_PHRASE>:N
                → (IS :N '*'?'* );

RULES OF QUESTION_INTRODUCER =
        COULD YOU TELL ME → ,
        WOULD YOU TELL ME → ,
        PLEASE TELL ME → ;
```

Although it is conceivable that there are an infinite number of ways to introduce a question in this manner, we have found only about six literal strings are actually used in our data base of dialogues. When we discover a new string we incrementally add a rule. When we have enough examples

to dectect a more genera) *farm* we *replace* the rules for <QUESTIONJNTR0DUCER> by a more elegant and general formulation. This approach allows us to process dialogues before we have a complete analysis of all possible sentence constructions, and it allows us to build a language analyzer based on actually occurring forms.

Notice that it is possible to make more than one analysis of any given sentence depending on what is being looked for. A poet might be interested in the number of syllables per word and. the patterns of stress. A "full" *analysis of* English must allow for this possibility, but it it clearly foolish to produce this kind of analysis for PARRY. Our analysis will be partial and idiosyncratic to the needs of our program. This is what is meant by idiolectic.

FILLERS

it is quite common for interviewers to introduce words of fittle significance to PARRY into the sentence. For example:

HELL, WHAT IS YOUR NAHE?

The "well" in this sentence serves no purpose in PARRY'S analysis, although it might to a linguist interested in hesitation phenomena. These fillers can be ignored The following rules accomplish this:

```
RULES OF SENTENCE -
        <F]LLERS>;F  <SENTENCE>:S  -  :S;

RULES OF FILLERS -
        WELL -  ,
        OK  -  ;
```

PUNCTUATION

Interviewers use little intra-sentence punctuation in talking to PARRY. When it is used it is often to seperate phrases that might otherwise be ambiguous. Example:

WHY WERENT YOU VERY CLOSE, FRANK

Here the comma clearly puts "CLOSE" in a different phrase from "FRANK". Punctuation, when used in PARRY'S rules, is generally enclosed in optional brackets ("[]"). This has the effect of separating phrases when punctuation is used, but not requiring full punctuation for the system to work, Example;

```
RULES OF SENTENCE -
     <SENTENCE>:S1  C.JtC  <SENTENCE_CONNECTOR>SC
        <SENTENCE>:S2
              - (CONUNCTION  SC :S1 :S2)|
```

CLICHES AND IDIOMS

The English we encounter in doctor-patient dialogues is made *up* of a great *number of cliches* and idioms, therefore we anticipate a large number of rules devoted to them. For example:

```
RULES OF TIME_PHRASES -
        A COUPLE OF <TIME_UNIT>;T AGO
        * (TIHE  (RELATIVE PAST)(REF PRESENT)  :T);

RULES OF TINE_UN1T -
        SECONDS  -  WITHIN CONVERSATION),
        MOMENTS  -  WITHIN CONVERSATION),
        DAYS  -  (BEFORE CONVERSATION DAYS):
```

REPRESENTATION CORRECTION

Intermediate results are often produced which are misleading in meaning or are in the wrong form for further processing. We, therefore, incorporate at various points rules which detect certain undesired intermediate results and convert them to the desired form. Example:

```
RULES OF CORRECT_FORM -
        (QUESTION  ...  (SENTENCE  ..,)) -
                (QUESTION. . . . . .):
```

UNKNOWN WORDS

Rules can be derived to handle words which were previously unknown to the system. For example:

```
RULES OF UNKNOWN_WORD -
        OR'.  )X  -  <NEW_WORD NAME  :X>,
        THE  :X  <VERB_PHRASE>:Y  *
                <NEU_U0RD NOUN  :X>,
        I  :X  YOU  -»  <NBJ_U0R0 VERB·  :X>i
```

Here "NEW_WORD" is a function which adds new words to the dictionary.

CONCLUSION

We are faced with the problems of natural language being used to interview people in a doctor-patient context. We have developed a language processing system which we believe is capable of performing in these interviews at a significantly improved level of performance compared to systems used in the past. We have developed techniques which can measure *performance in comparison with* the ideal of a real human patient in the same context,"[5,7] We are designing our system with the realization that a long period of development is necessary to reach desired levels of performance, This is a system that can work at a measured level of performance and be improved over time with new rules having minimum interaction with those already existing. Our system is designed so that a complete analysis of every word or phrase of an utterance is not neceesary.

The basis of this system is a rewrite interpreter which will automatically merge new rules into the set of

already existing rules so that the system will continue to handle sentences which It handled in the past.

REFERENCES

'Carroll, J.B. Language and Thought. Prentice-Hall, Englewood Cliffs, New Jersey, p. 59.

[2]Colby, K.M., Watt, J. and Gilbert, J.P. A computer method of psychotherapy. Journal of Nervous and Mental Disease, 142,148-152,1966.

[3]Colby,K.M. and Enea,H. Heuristic methods for computer understanding of natural language in context restricted on-line dialogues. Mathematical Biosciences,I, 1-25,1967.

[4]Colby, K.M., Hilf, F.D., Weber, S., and Kraener, H. Turing-like indistinguishability tests for the validation of a computer simulation of paranoid processes. Artificial Intelligence.3,199-221,1972.

[5]Colby, *KM.* and Hilf, F.D. Multidimensional analysis in evaluating the adequacy of a simulation of papamoid processes. Memo AIM-194. Stanford Artificial Intelligence project, Stanford University.

[6]Enea, K MLISP, Technical report no. CS-92, 1968, Computer Science Department, Stanford University.

[7]Rubenstein, A.H. and Haberstroh, C. J,, Some Theories of Organization, Dorsey Press, Homewood Hl.,1960, p. 232.

[8]Schank, R.C., Tesier, L and Weber.S. Spinora ii: Conceptual case-based natural language analysis. Memo AIM-109, 1970, Stanford Artificial Intelligence Project, Stanford University.

[9]Simmons, R.f. Some semantic structures for representing English meanings. Preprint, 1970, Computer Science Department, University of Texas, Austin.

[I0]Smith, D.C., MLISP, Memo AIM-135, 1970, Stanford Artificial Intelligence Project, Stanford University.

[11] 'Weizenbaum, J. Eliza- a computer program for the study of natural communication between man and machine. Communications of the ACM, 9,36-45,1966.

[12]Wilke, Y.A. Understanding without proofs. (See this volume).

[13]Winograd, T. A program for understanding natural language. Cognitive Psychology,3,l-l91,1972

[14]Woods, W.A. Transition network grammars for natural language *analysts.* Communications *of the* ACMJ 3,591-606,1970.

MECHANISM OF DEDUCTION IN A QUESTION ANSWERING SYSTEM WITH NATURAL LANGUAGE INPUT

Makoto Nagao and Jun-ichi Tsujii
Kyoto University, Dept. Elec. Eng.
Kyoto, Japan

ABSTRACT

We have constructed a deductive question answering system which accepts natural language input in Japanese. The semantic trees of aseertional input sentences are stored in a semantic network and interrelationships —conditional, implicational, and so forth— are established among them. A matching routine looks for the semantic trees which have some relations to a query, and returns the mismatch information (difference) to a deduction routine. The deduction routine produces sub-goals to diminish this difference. This process takes place recursively until the difference is completely resolved (success), or there is no other possibility of matching in the semantic network (failure). Standard problem solving techniques are used in this process. As the result the system is very powerful in handling deductive responses. In this paper only the part of the logical deduction is explained in detail.

DESCRIPTIVE TERNS: question answering, deduction, natural language, semantic network, problem solving.

I INTRODUCTION

There are a few deductive question answering systems using natural language, almost all of which use logical expressions, especially the first order predicate calculus expression, as an intermediate language. However systems which use formal logics have problems:

(1) Syntactic and semantic analyses of natural language input are necessary to transform the input to logical expression without ambiguity.
(2) The axiom set must be clearly defined and must not be contradictory.
(5) Predicates and variables must be fixed beforehand. This is a problem for the system's expansion. Also this prevents mixing the first and higher order predicate calculus systems.
(4) Deduction using the resolution principle is cumbersome. Usually question answering does not require a deep deductive process.
(5) Good quality of natural language output is very . hard to obtain from a logical expression.

To avoid the above problems we have used a kind of semantic representation of natural language sentences as an intermediate expression. pur sustern has the following characteristic features.

(1) The question answering system is a composite of subsystems for language analysis, deduction, and language generation.
(2) The parsed trees of sentences are permitted to have some ambiguities. Ambiguities are resolved in the process of logical deduction,
(3) During the question answering process, the deduction ability is increased and the area which the system can deal with is also expanded. The deduction ability of a system depends on how many theorems the system can use, and on how efficiently it can deal with them. We have constructed a system in which the available theorems increase during the question answering process.
(4) Facts can play the role of theorems. We think the distinction between facts and theorems is not clear enough. A statement can be used as a theorem at one time and as a fact at another time. For example,

A human is an intelligent animal,

plays the role of a theorem to answer

Is Smith intelligent ?

because Smith is an instance of a variable 'human'. On the contrary it plays the role of a fact to the question

Is a man an animal ?

because 'a human' is treated as an instance of a variable 'man'.

In our system the assertions given by a user, which correspond to facts in usual systems, can play the role of theorems. This is accomplished by allowing a higher concept term to be a variable to its lower concept term. There is no distinction between them, and both facts and theorems have the same structures in the data base. This is the most significant character of the system we have developed.

(5) In order to deal with a large data base, the system has a well organized data structure and relevant information to a question is accessed by a technique of indexing and retrieval,
(6) The deduction process is similar to that of humans. It allows introducing many heuristics into the deduction process.

In this paper the details of deduction subsystem alone are explained. The other two subsystems will be published elsewhere in the near future,

II SYSTEM ORGANIZATION

A block diagram of our system is shown in Fig. 1. The internal data base of the system is divided into two parts:

(1; semantic representations (semantic trees) of input sentences.

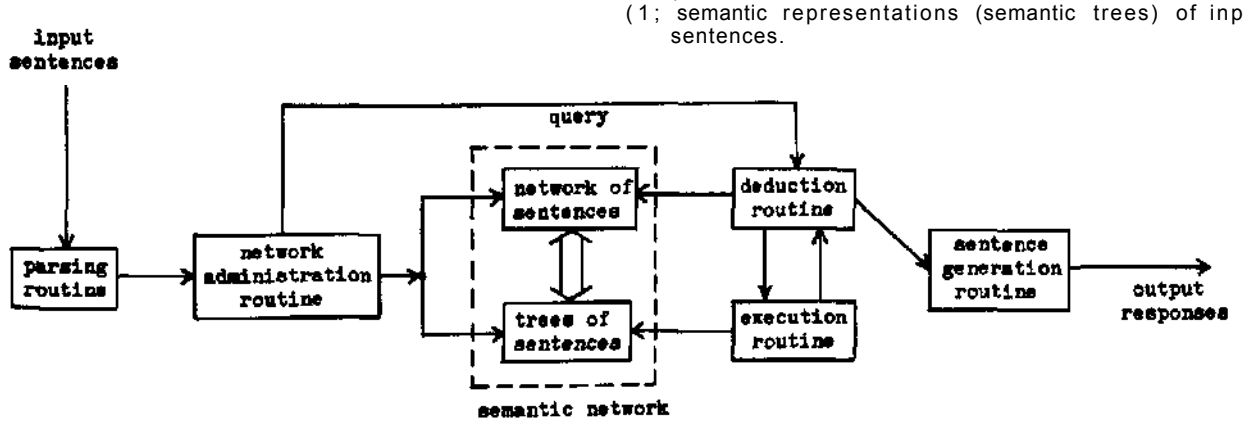

Fig. 1 Organization of the system.

(2) network (mutual connection) of (1).
The mutual connection consists of interrelationships such as conditional, implicational, and so forth. An input sentence is analyzed into a semantic tree, and it is read into the semantic network if it is a*n* assertion and is not in the network yet. Thus knowledge accumulates in a very natural way in the question answering process. An inverted file of keywords makes it easy to extract information relevant to the question.

The parsing routine performs syntactic and semantic analyses of an input query sentence, and *produces* the parse tree. A network administration routine accepts the tree and relates it to the semantic network which contains sentences already accepted.

To accomplish a deduction, there are two main parts: the execution routine and the deduction routine. The execution routine, which plays the central role in the deduction process, searches through the network for sentences relevant to the current goal and matches them one by *one* against it. The deduction routine manages the global information in the problem solving process such as goal-subgoal relationships, variable bindings (for example the word 'man' is bound to the word •Smith'), and so forth. This routine also directs the execution routine to determine which sentence must be verified first.

Ⅱ KNOWLEDGE STRUCTURE

3.1 Semantic Trees.

We have applied a kind of dependency analysis to the input Japanese sentences. A noun modified by an adjective is transformed into a kernel sentence having another kernel sentence related to the noun. The sentence

KINBEN MA WITO WA SEIKO SURU
(A diligent man will succeed.)

is divided into two sentences like

HITO WA SEIKO SURU
(A man will Bucceed.)

and

HITO WA KINBEN DA
(A man is diligent.)

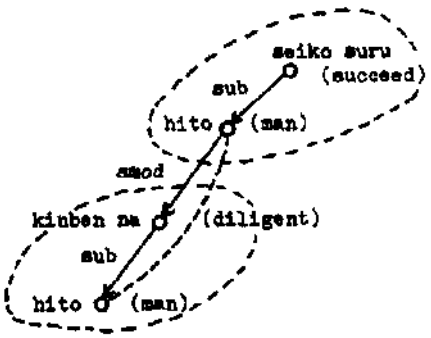

Fig. 2 Kinben na hito wa seiko suru.
(A diligent man will succeed.)

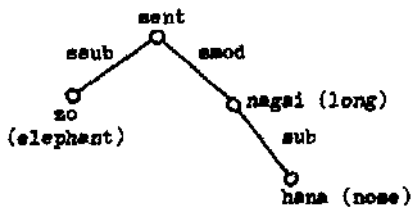

Fig. 3 Zo wa hana ga nagai.
(Elephant has a long nose.)

The parsed tree stfueture of this sentence is shown in *Fig. 2.*

Some sentences in Japanese have two possible subject phrases, that is, one which contains the reference particle 'GA' and the other which contains 'WA*. We consider the relational phrase with the particle 'WA' as indicating what the sentence talks about; the phrase with 'GA' is the subject phrase corresponding *to* the predicate in the sentence.

ZO WA HANA GA NAGAI
(Elephant has a long nose.)

is a typical example. Its literal translation is " As for elephant the nose is long." The tree structure of it is shown in Fig, 3.

Sentences connected by AND or OK are represented in the tree structure as shown in Fig. 4.

A *sentence* which contains upper concept terms replaceable by their lower concept terms is considered as a theorem available to prove a statement which has the lower concept terms in it. *So* upper-lower concept relationship among words plays an important role in our system. The input sentence in the form of " A WA B DA" meaning A is a lower concept of B, and B is an upper concept of A, has a special structure to express the relationship clearly. " NINGEN WA KASHIKOI DOBUTSU DA" (A man is a*n* intelligent animal.) is parsed as shown in Fig. 5.

Properties of sentences are attached to the top node of the parsed tree structure. The properties we treated are potential, active, passive, subjenctive, tense, and so forth. The assertion sentence is regarded as true, so that a sign T is given to the property part of the parsed tree. The signs F and U in the property part indicate false and undetermined respectively.

3.2 Semantic Network.

The network is constructed in the following way.

(1) In the case of an assertion sentence S_{nn}, it is stored in the form shown in Fig. 6a. T.

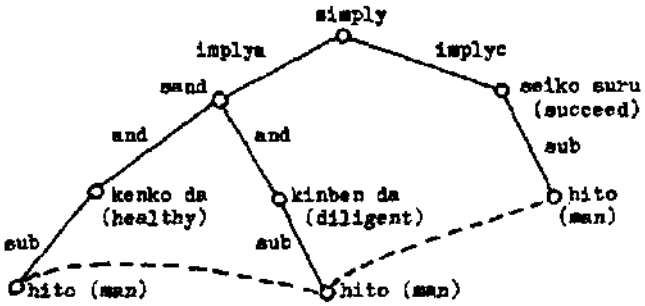

Fig. 4 Kenko de kinben na hito wa seiko suru.
(A man who is healthy and diligent will succeed.)

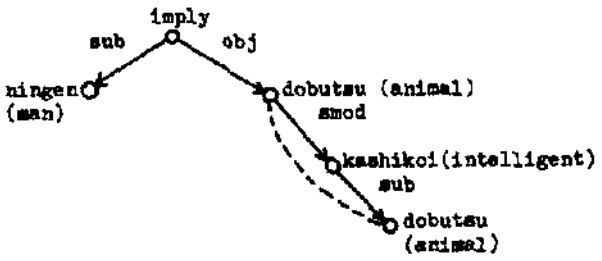

Fig. 5 Ningen wa kashikoi dobutsu da.
(A man is an intelligent animal.)

(2) In the case of a negation sentence, schematically written as 'not S_2 it is stored in the same form ae Fig. 6a, but the property part is written as F.

(3) If a sentence is -If s_1, then S .', it is stored in the form shown in Fig. 6b.

(4) If a sentence is 'Because S_1, S_2. it is stored in the form shown in Fig. 6c.

(5) If the sentences S_1 and S_2 in <1)--(4) are found in the semantic network, they are not stored newly, but the stored ones are used. For example the following sentences are stored in the network as shown in Fig. 6d.

Because S_1, S_2.

If S_1, then S_3.

In this case because S_1 is asserted as true, S_3 is also true.

The network and parsed trees have the following internal constructions.

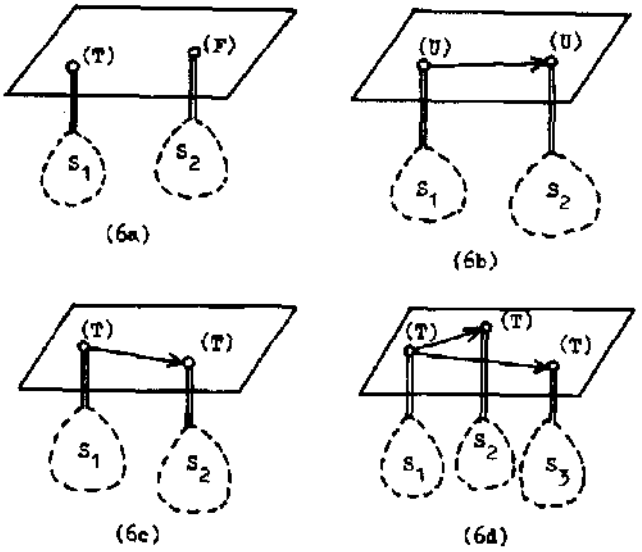

Fig. 6 Relations in semantic network.

{1) Branches in the network and trees are bi-directional for flexible transformation and for efficient search in the deduction process.

(2) Words are not stored in nodes of the parsed trees but by a pointer to the lexical entry of the word (Fig. 7).

(3) The lexical entry of a word, called NLIST, contains not only lexical information about the word, but also a list of sentences (pointers to the entries of the sentences in SLIST) which contains the word. NLIST is a kind of inverted file of keywords.

CO The node of the network is indicated by a pointer from a table, called SLIST, which contains information about the sentence. The information of whether the sentence is true (T), false (P), or undetermined (U), and so forth is stored in this list.

(5) Different nodes in the network correspond to different sentences. As a result, information about a sentence can be retrieved from a single node in the network.

IV EXECUTION ROUTINE

Among many intellectual abilities of humans, we have implemented in this study the deduction ability baaed on the use of "the law of substitution' and 'the law of implication.' This is realized by the execution routine and the deduction routine. The execution routine tries to match a sentence structure against another one, regarding an upper concept as a variable over its lower concepts. The deduction routine produces subgoals and tells the execution routine which sentence must be verified first. The execution routine searches through the network for the sentences which are equivalent to the goal sentence given by the deduction routine. It consists of three main parts: keyword search, matching, and resolving differences.

4.1 Keyword Search.

The system has an inverted file of words called NLIST. By using this file, the execution routine takes out the sentences which contain words in the goal sentence. These selected sentences are presumed to be relevant to the current sentence.

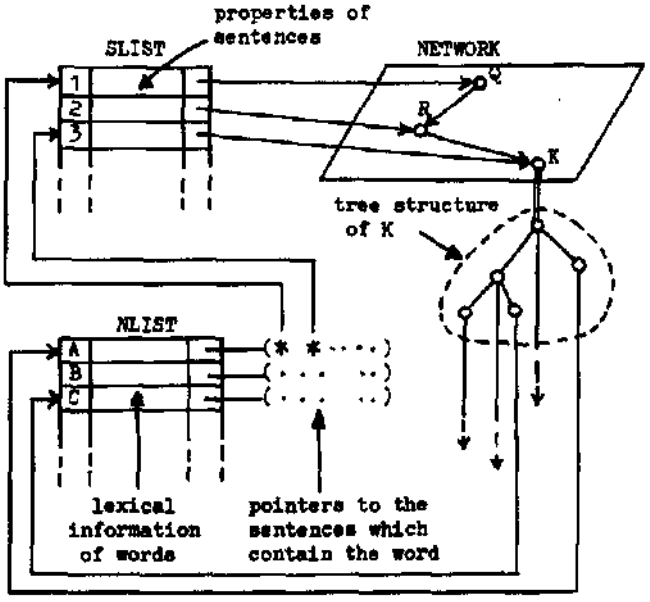

Fig. 7 Internal data base structure.

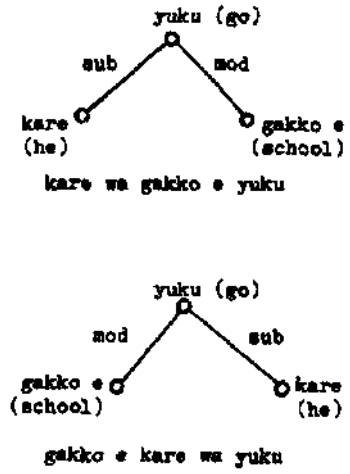

Fig. 8 Change of word order.

4.2 Hatching Method

The matching algorithm is constructed so that two parsed trees which are different in the sequence of branches <Fig. 8) will be matched successfully by the branch labels on the parsed trees. Matching between two parsed trees fails for various reasons. The causes of mismatch, named differences, are classified into the following four classes.

(1) N-difference: The words which are attached to the corresponding node are different in the two sentences. Fig. 9a shows an example, where the difference is expressed as (N (*C *D)). *C shows the pointer to the node C.

(2) S1-differenee: One structure (first argument) has extra branches which the other does not have. Fig. 9b shows an example of this category, abbreviated as CS1 ((*R4) -B)), which shows the branch *R4* is the extra one.

(3) S2-difference: One structure (second argument) has extra branches. Fig. 9c is an example and this difference is shown by (s2 (*C (*R5))).

(4) SO-difference: Both structures have extra branches. An example is shown is Fig. 9d.

The matching subroutine tries to match its first argument against its second one. If the matching succeeds, the subroutine returns 'success' to the deduction routine. If not, it returns the differences.

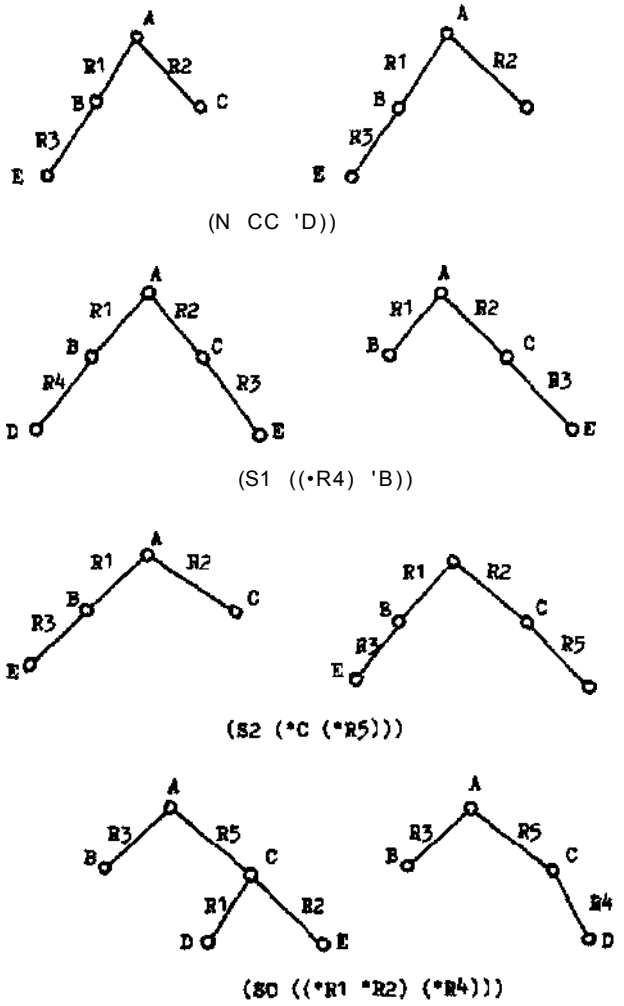

Fig. 9 Differences in matching.

4.3 Resolving Differences.

The execution routine first picks up sentences expected to be relevant to the given sentence by using NLIST, and then tries to match them against the given sentence. If the same sentence is stored in the data base, the execution routine picks it up and the matching ends in success. If there is no complete match, but a difference, N-routine or S-routine is activated according to the kind of difference to resolve the difference.

(1) N-routine

An N-routine arises from mismatch of words. Let us suppose that the sentence

TARO WA SEIKO SURU
(Taro will succeed.)

is what the deduction routine tells the execution routine to prove, and the sentence

NINGEN WA SEIKO SURU
(A man will succeed.)

is stored in the data base. The matching between these does not succeed and the difference is (N (*TARO *NINGEN)). This difference is transferred to N-routine and the routine tries to check whether the word TARO is a lower concept of NINGEN (man) by searching through the network for the sentence 'TARO WA NINGEN DA', which means 'TARO is a lower concept of NINGEN.' If such a sentence is found, NINGEN can be looked upon as a variable which can take the value TARO, and then the difference is resolved. This is considered as the process of substitution. By this process the system can deduce specific facts from generalized knowledge.

N-routine basically searches the sentence 'A WA B DA', which means 'A is B', in order to resolve the difference (N (*A *B)), but many sentences in the network are in such forms as 'A WA b NA B DA', which means 'A is B modified by b', and 'aNA A WA B DA', which means 'A modified by a is B'. The differences to be resolved also take the forms of (N (*(a NA A) *B)) and (N (*A *(b NA B))). Four cases are possible.

(a1) Difference : (N (*(a NA A) *B))
In the data base : A WA B DA
In a logical representation,
the goal to be proved is $a(x) \wedge A(x) \rightarrow B(x)$
the fact in the network is $A(x) \rightarrow B(x)$
and the difference is resolved immediately.

(a2) Difference : (N (*A *(b NA B)))
In the data base : A WA B DA
In a logical representation,
the goal to be proved is $A(x) \rightarrow b(x) \wedge B(x)$
the fact in the network is $A(x) \rightarrow B(x)$
In this case whether A satisfies the condition b or not is produced as a subgoal.

(b1) Difference : (N (*A *B))
In the data base : A WA b NA B DA
In a logical representation,
the goal to be proved is $A(x) \rightarrow B(x)$
the fact in the network is $A(x) \rightarrow b(x) \wedge B(x)$
So the difference is resolved.

(b2) Difference : (N (*A *B))
In the data base : a NA A WA B DA
In a logical representation,
the goal to be proved is $A(x) \rightarrow B(x)$
the fact in the network is $a(x) \wedge A(x) \rightarrow B(x)$
In this case a subgoal is produced.

(2) S-routine

S-routine resolves S1-, S2-, and SO- differences. These differences arise from mismatch of branches. S-routine is given two different sentence structures, one is called S-structure and the other is called T-structure. Using grammatical rules (especially transformational rules), this routine transforms the S-structure into several transformationally equivalent structures, and matches them against the T-structure. At present not so many transformational rules are prepared.

Fig. 11 is an example. If the matching succeeds, the two structures, S-structure and T-structure, are equivalent and the difference is resolved.

V DEDUCTION ROUTINE

The deduction routine controls the whole of the deduction process. This routine has a global knowledge of the process. This knowledge contains the goal-subgoal organization, variable binding and so forth, The deduction routine tells the execution routine which sentence must be verified and which sentence, if the first trial fails, has to be verified next.

5.1 Goal Organization

The deduction method in our system takes a question Q as a .goal and tries to verify it by means of matching it with the sentences stored in the network. If the trial fails, the deduction routine searches through the network for such sentences as P-*Q. Those sentences P's, if any, are considered as subgoals to accomplish the previous goal. In the same manner sub-subgoals are produced to accomplish the subgoals. As the process advances, many goals are produced hierarchically. An AND-OR tree structure is used to remember the hierarchically organized relationships among goals.

Subgoals are created in various cases.

(1) If a goal sentence G can not be determined to be true or false, subgoals are created by means of searching through the network for the sentences which are antecedents of G.
(2) In the same case of (1), the negations of consequences of G are taken as subgoals. If they are proved to be true, the sentence G is determined to be false.
(3) If the matching between two parsed trees is incomplete, subgoals to diminish the mismatches are created.

In addition to these cases, subgoals are also produced when a goal is divided into several subgoals. For example 'KARE WA KINBEN DE SHOJIKI DA' (He is diligent and honest) is divided into 'KARE WA KINBEN DA' (He is diligent), and 'KARE WA SHOJIKI DA' (He is honest).

The goals are tried one by one, and when there remains no goal, the deduction process stops with a failure message. A goal which has several subgoals will succeed or not, depending upon whether the subgoals will succeed or not. A goal keeps some information for itself. For example it has the information of whether it is an AND-type or an OR-type. Depth of goal shows the depth between the top-goal (that is, a question given by a user) and the present goal. The depth of the top-goal is 0 and the depth of the immediate subgoal is 1.

The deduction routine chooses a goal, the depth of which is the smallest of all, and tells the execution routine to verify it. The indicators such as KOTEI (positive assertion), HITEI (negative assertion), MATCH (to be matched) and so forth show the effects of the goals' results to be transferred to their previous goals. KOTEI (HITEI) shows that if this goal succeeds, the sentence corresponding to its previous goal is proved to be true (false). The subgoals which are produced in order to resolve the mismatch between two parsed trees have the indicator MATCH.

5.2 Variable Binding.

To use the law of substitution is one of most important abilities in this system. This is carried out by considering an upper concept as a variable over its lower concepts. A word behaves as a constant when it is a lower concept of another word, and as a variable when it is an upper concept of another word. We do not introduce unary predicates such as'human(x)', 'anlmal(x)', which are usually used in the predicate calculus system in order to restrict the range of variables.

We regard all words as variables which have their own domains of values. We illustrate this by the following example.

(1) HITO GA KENKO NARA-BA HITO WA SEIKO SURU
(If a man is healthy, the man will succeed.)
(Q) Smith WA SEIKO SURU KA ?
(Will Smith succeed ?)

The system searches through the network to find out the sentence (1) which is expected to answer the given question. The matching between the consequent part of (1) and the question fails at first. The cause of mismatch is N difference between 'Smith* and 'HITO (man)'. N routine is called to find out that HITO is an upper concept of Smith, which is proved by the information 'Smith is a man.' in the network. Thus a subgoal, the antecedent of (1), in which HITO is replaced by Smith is produced, that is, 'Is Smith healthy ?'. As the deduction process proceeds, several such bind conditions are produced. Each goal must be tried taking into consideration the related bind conditions produced during the former process.

The deduction routine has a stack to remember these conditions. This stack is illustrated in Fig, 10. Each goal has a pointer to this stack and the routine can retrieve the corresponding bind condition of a goal. If a goal fails, then the bind condition generated during the trial of the goal is abandoned. On the other hand if a goal succeeds, its condition is memorized for use in the succeeding process.

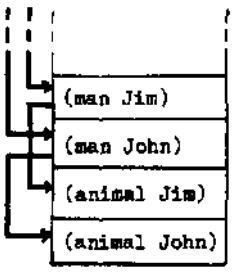

Fig. 10 Stack for variable binding.

VI COMPARISON WITH THE SYSTEMS USING PREDICATE CALCULUS

Those systems which use predicate calculus translate the input into a predicate calculus formula, store it in the data base, and use a universal method of deduction such as the resolution method. In those systems common subexpressions appearing in different sentences are stored as many times as they appear in different logical formulas. This is not efficient. In our system the same subexpressions are stored only once and their relations to the other parts of sentences are stored by links. So these interrelationships can be utilized in the deduction process. Especially when the system deals with a great amount of data and only a relatively small portion of the data has a direct relation to the given question, the quick access to these related expressions is very important in the deduction process.

Which sentences or formulas are available for the current problem needs to be recognized easily, and to do this, a well organized data base is necessary. It is tempting to try to incorporate the use of property lists to speed up resolution. For example one may find it useful for each object symbol c to have access to a chained list of all literals or clauses where c occurs.

A difficult but more important problem is to recognize how a meaningful unit is related to another unit. It is desirable for the data base to contain information about the interrelationships among the meaningful units. In our system the deduction procedure can retrieve from a node those sentences which have some

relation to the sentence corresponding to the node.

Another is that disambiguation is done not only in the parsing phase but also in the deduction phase. For example, the sentence 'A NO B WA ... ' may have more than four different structures in deeper levels, according to the words A and B. That is:

KARE NO KANE — the money which he has
(he) (money)
KYOSHI NO KARE — he, who is a teacher
(teacher) (he)
KYONEN NO SENKYO — the election which was taken place last year
(last year) (election)

The parsing and translation program in predicate calculus system must choose one of these structures at the input and parsing stage, because predicate calculus formulas never permit ambiguous expressions. But it is almost impossible to classify each word into a certain semantic category, and to decide which of the above structures is proper to the sentence according to the information that the word A belongs to a certain category and B belongs to another.

In our system, 'A NO B WA ... ' is stored as shown in Fig. 11. The ambiguity is left in its structure. The matching routine transforms the sentence into several different structures by using grammatical knowledge, and tries to match them one by one against the object structure. All of them except one correct structure may not match against it. Thus ambiguous structures are resolved during the deduction process.

This is also one of the excellent features of using semantic structures of sentences which permit ambiguous structures as an internal data representation.

VII EXAPLES

Example 1

Input sentences:

HITO WA KENKO DE KINBEN NARA SEIKO SURU.
(If a man is healthy and diligent, the man will succeed.)
HITO WA sportsman NARA KENKO DESU.
(If a man is a sportsman, the man is healthy.)
Jim WA sportsman DESU.
(Jim is a sportsman.)
Jim WA KINBEN DESU.
(Jim is diligent.)
Jim WA KASHIKOI HITO DESU.
(Jim is a clever man.)

Question given to the computer

Jim WA SEIKO SHIMASU KA ?
(Will Jim succeed ?)

Responses from the computer

Jim WA KENKO DE KINBEN KA ? (Is Jim healthy and diligent ?)
Jim WA KENKO KA ? (Is Jim healthy ?)
Jim WA sportsman KA ? (Is Jim a sportsman ?)
Jim WA KINBEN KA ? (Is Jim diligent ?)
HAI, Jim WA SEIKO SURU. (Yes, Jim will succeed.)

These outputs except the last are the intermediate ones from the computer, to which no answers are necessary.

Example 2

Input sentences

Jim was killed by John.
A man-A who killed a man-B is punished.
Jim is a man.
John is a man.

Question

Is John punished ?

Responses from the computer

Did John kill a man-B ?
Yes, John is punished.

Example 3

Whale bears a child.
An animal which bears a child is a mammal.
If an animal is a mammal, the animal is a vertebrate.
A vertebrate has a backbone.

Question

Has whale a backbone ?

Responses from the computer

Is whale a vertebrate ?
Is whale a mammal ?
Does whale bear a child ?
Yes, whale has a backbone.

In these examples intermediate responses are to show the deduction processes, which do not need answers from a man.

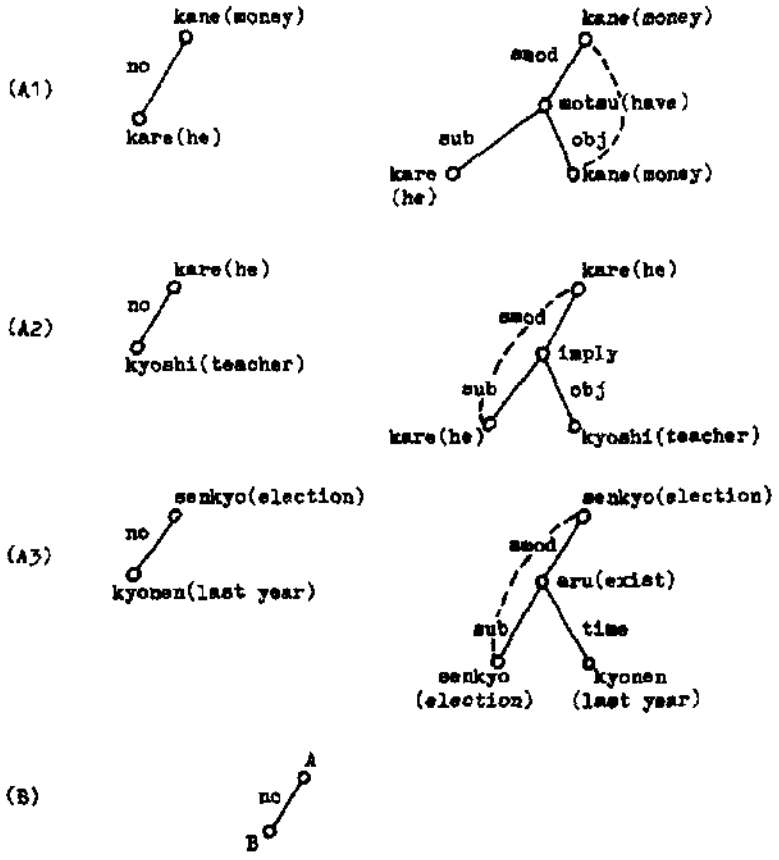

Fig. 11 (A) Several possible deep structures for 'A no B'.
(A1) the money which he has.
(A2) he, who is a teacher.
(A3) the election which took place last year.
(B) Internal representation in our system.

REFERENCES

(1) S.C. Shapiro, A Net Structure for Semantic Information Storage, Deduction and Retrieval, AI Conf. 71, 1971, p.512
(2) E.J. Sandewall, Formal Methods in the Design of Question-Answering System, J. Art. Int. 1972, p.237
(3) M.R. Quillian, The Teachable Language Comprehender; A Simulation Program and Theory of Language, CACM, vol.12, No.1, 1972, p.456
(4) B. Raphael, C. Green, The Use of Theorem Proving Techniques in Question Answering System, JACM, 1968, p.169