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 system 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.

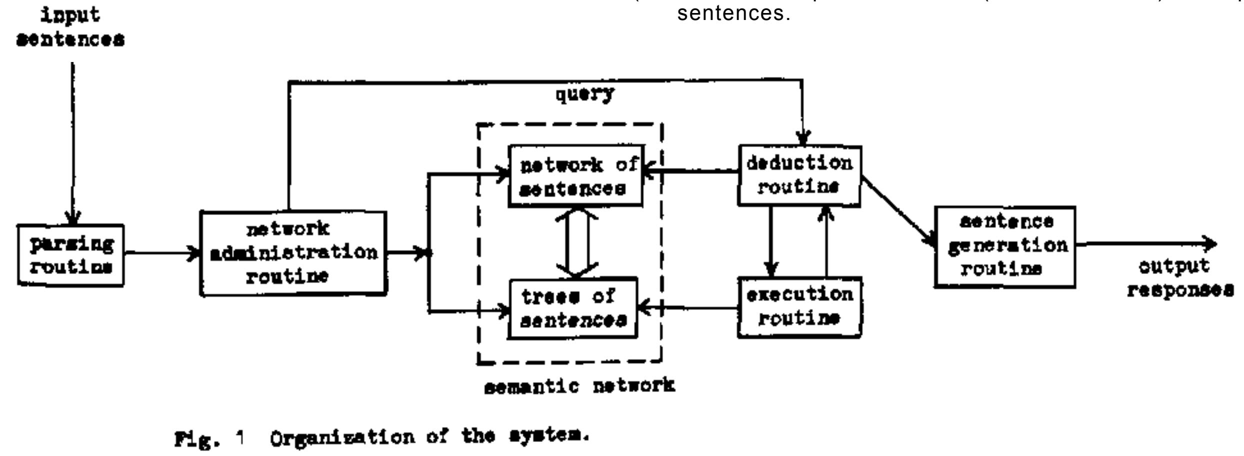


Fig. 1 Organization of the system.

(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.

III 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.)

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_i, it is stored in the form shown in Fig. 6a.

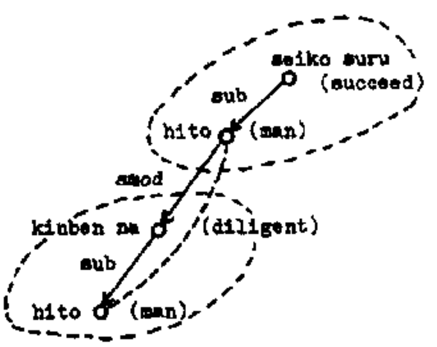


Fig. 2 Kinben na hito wa seiko suru.
(A diligent man will succeed.)

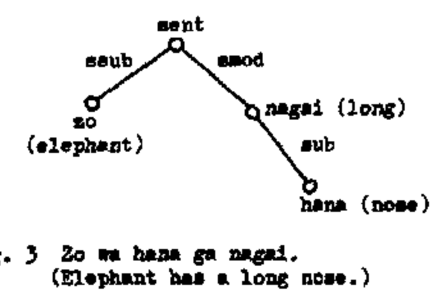


Fig. 3 Zo wa hana ga nagai.
(Elephant has a long nose.)

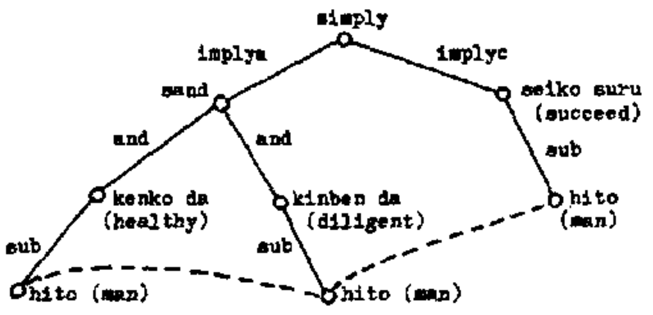


Fig. 4 Kenko de kinben na hito wa seiko suru.
(A man who is healthy and diligent will succeed.)

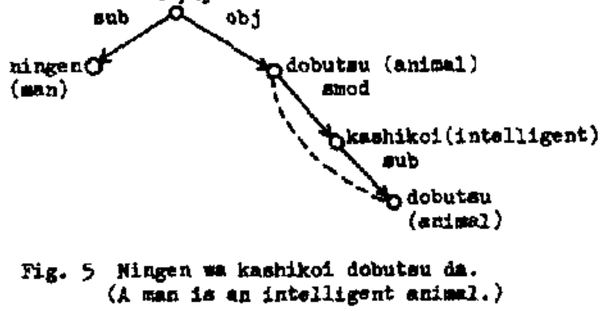


Fig. 5 Ningen wa kashikoi dobutsu da.
(A man is an intelligent animal.)

(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. 5

The network and parsed trees have the following internal constructions.

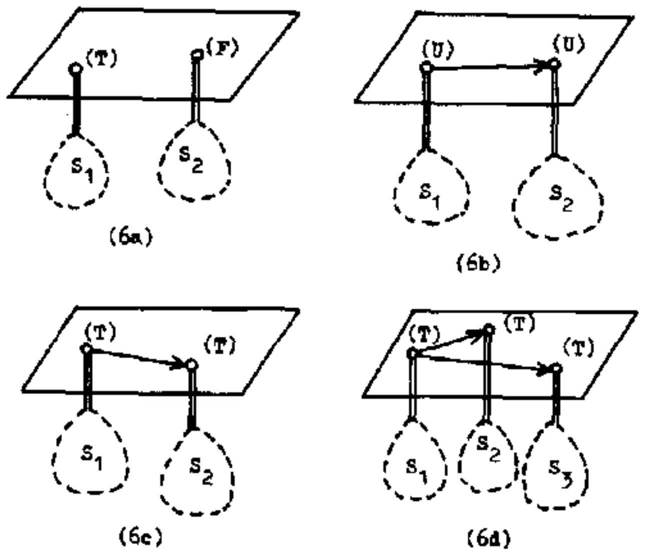


Fig. 6 Relations in semantic network.

{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.

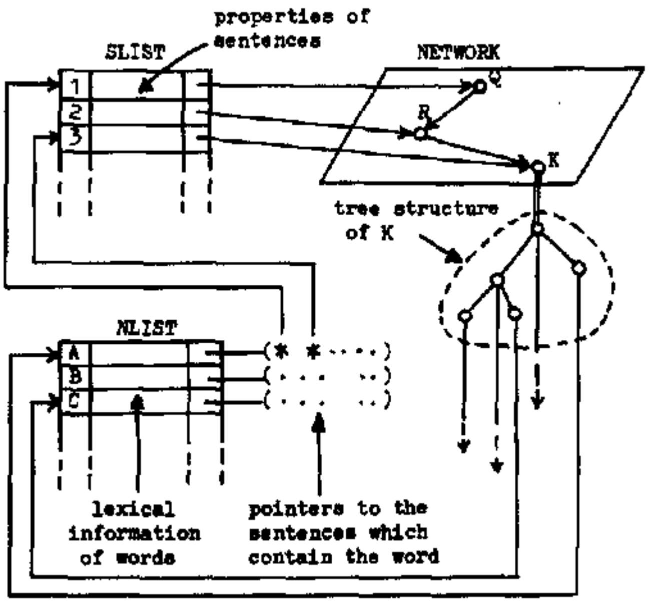


Fig. 7 Internal data base structure.

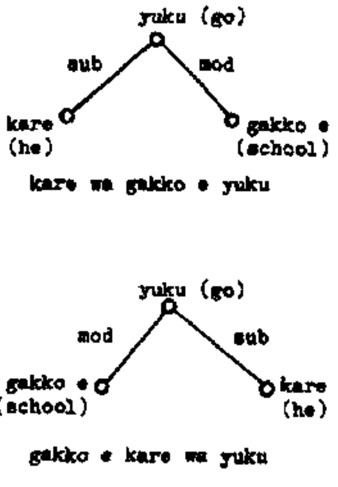


Fig. 8 Change of word order.

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.

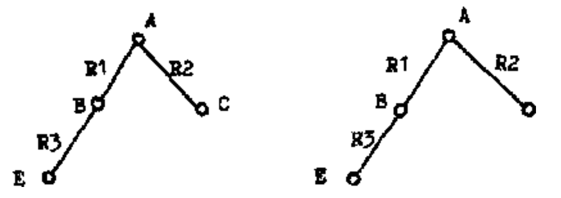
(1) N-difference: The words which are attached to the corresponding node are different in the two sentences. Fig. 9a shows an example, where the difference is expressed as (N (*C *D)). *C shows the pointer to the node C.

(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.

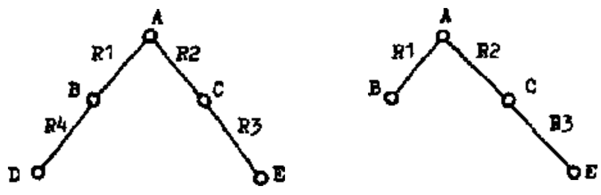
(3) S2-difference: One structure (second argument) has extra branches. Fig. 9c is an example and this difference is shown by (s2 (*C (*R5))).

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

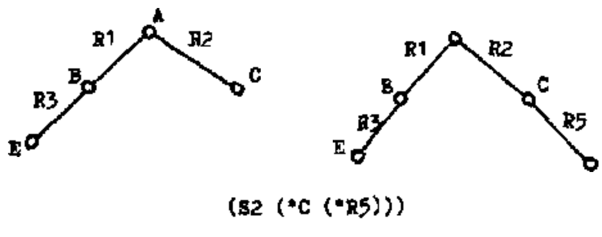
The matching subroutine tries to match its first argument against its second one. If the matching succeeds, the subroutine returns 'success' to the deduction routine. If not, it returns the differences.



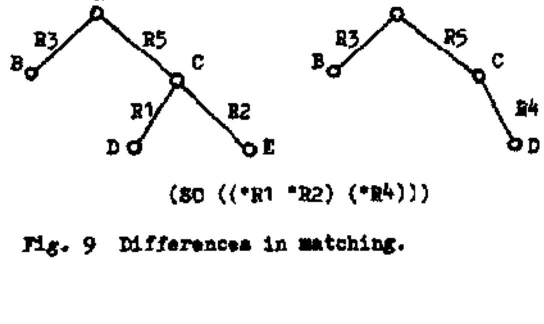
(N CC 'D))



(S1 ((•R4) 'B))



(S2 (*C (*R5)))



(SO ((*R1 *R2) (*R4)))

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.

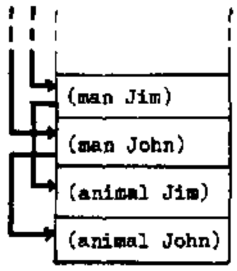


Fig. 10 Stack for variable binding.

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

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.

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

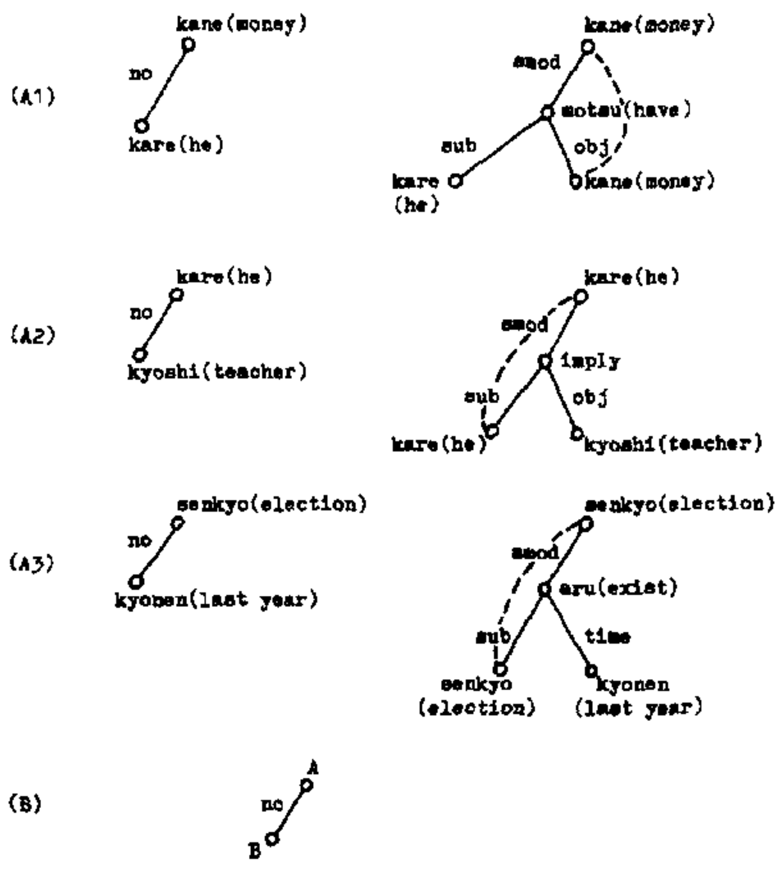


Fig. 11 (A) Several possible deep structures for 'A no B'.
(A1) the money which he has.
(A2) he, who is a teacher.
(A3) the election which took place last year.
(B) Internal representation in our system.