A COMPUTER MODEL

OF

NATURAL LANGUAGE UNDERSTANDING FOR JAPANESE

Jun-ichi Tsujii

Department of Electrical Engineering

Faculty of Engineering

Kyoto University

March 1978

A COMPUTER MODEL

OF

NATURAL LANGUAGE UNDERSTANDING FOR JAPANESE

Jun-ichi Tsujii

Department of Electrical Engineering

Faculty of Engineering

Kyoto University

March 1978

A COMPUTER MODEL OF NATURAL LANGUAGE UNDERSTANDING FOR JAPANESE

Jun-ichi Tsujii

ABSTRACT

The research described here is concerned with the development of a natural language understanding system. Various problems encountered during the development of the system are discussed.

A new programming language PLATON has been developed to simplify the writing of natural language analysis programs. It has facilities for pattern matching and flexible backtracking. Flexible interactions between a syntactic analysis procedure and other components, for instance semantic and contextual analysis procedures and so on, can be easily realized by using the pattern matching facility. The backtracking facility permits fairly complicated nondeterministic programs to be written in a simple manner.

A program for analyzing Japanese sentences has been developed by using PLATON. The program utilizes not only syntactic constraints but also semantic and contextual information to disambiguate fairly complex sentences. The idea of "case" proposed by C.J.Fillmore is adopted and modified to represent the meaning of verbs and nouns. Contextual information is represented in the form of semantic networks. By utilizing these representations, the program can appropriately analyze long Japanese noun phrases concatenated by the postposition "NO", and long Japanese sentences with embedding sentences. Omitted words and anaphoric expressions can also be analyzed successfully.

As a higher level of representation, another form of semantic

network (called S.N.) is provided. While the meaning descriptions in the dictionary are organized in a form convenient for sentence analysis, the S.N. is designed to facilitate deduction and problem solving. The S.N. provides a framework in which various kinds of knowledge, such as procedural knowledge, knowledge about external data bases, algorithmic knowledge etc., can be naturally integrated. Examples of problem solving using the S.N. are provided.

Designers of language understanding systems often bypass the problem of morphological analysis. This is unrealistic for Japanese, which has no definite word delimiters and uses compound words with a high frequency. Since the writing system of Japanese is quite different from those of European languages, we have had to devise our own morphological analysis procedure for Japanese. A large computerized Japanese dictionary is used for the procedure. The data structure and some efficient access methods are discussed.

ACKNOWLEDGMENTS

The author would like to express his sincere thanks to professor Makoto Nagao of Kyoto University for the supervision and continuous encouragement to complete this thesis. Not only he guided the author to the present study but also gave him many helpful suggestions and constructive comments during the course of this study. He also critically read the manuscript of this thesis and gave the author accurate comments to be sincerely appreciated.

The author also wishes to thank Professor Toshiyuki Sakai. He initially guided the author to this research area.

The author is grateful to Mr. Kazutoshi Tanaka, Mr. Akira Terada, Mr. Akira Yamagami, Mr. Shuji Tatebe and the other members of Professor Nagao's research group for their useful discussions with the author. Mr. Tanaka contributed significantly to the content of Chapter III and Chapter IV. Concerning Chapter V, Mr. Terada helped the author very much in designing the S.N.. Mr. Yamagami and Mr. Tatebe helped the author in the implementation of the morphological procedure described in Chapter VI.

Dr. Michael J. Freiling read the manuscript of the thesis and corrected many errors in English.

The author is also grateful to Miss Setsuko Akiyama and Miss Hidemi Funagoshi for the preparation and typing of this thesis.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

I-1 Introduction

Why are we interested in language processing by computer ?

People are interested in human language activities for various reasons and the researchers in many established disciplines are engaged in uncovering various aspects of human language activities through their own research methodologies. Linguists have striven to uncover syntactic or semantic features common to all of the world's languages. They concentrate their research efforts on constructing a theoretical framework in which the grammar of every language in the world can be well explained. One of the main goals of psychological studies is the experimental and theoretical understanding of human cognitive processes, and language activities play a central role in those processes. Psychologists investigate the internal mechanisms of human cognition through observing how a child learns his native language and how people communicate with each other by means of language.

Why however are computer scientists interested in natural language ? Furthermore, from what viewpoints and by what methodology should they study natural language phenomena ?

From the practical point of view, it is desired that a computer be able to understand and process natural language because of the following reasons :

1. *Natural language is the most convenient and flexible media for man-machine communications.*

In the near future we can expect a vast increase in the range

of computer applications and a corresponding increase of the effect of computers on our daily lives. All sorts of people, not only computer specialists but also non-specialists such as researchers of various fields of natural and social sciences, managers of various enterprises, and even housewives will enjoy the benefits of computer technology.

One of the main objectives of the attempt to develop systems which can understand natural language is to make it easier for ordinary people to interact with a computer system directly and fluently.

2. *Natural language is one of the most important types of data which may be processed and stored by computer.*

By the benefit of the recent development of technologies, we have come to be able to process various kinds of data by computer. However, there still remain some important types of data which cannot be processed freely by a computer. Natural language texts are one of these types. It is still not possible to automatically translate texts from one natural language into another. Though we can extract spectral information easily from arbitrary wave forms by computer, we cannot extract key words automatically from natural language texts.

One of the main objectives of developing a system which can process natural language data is to offer people the capability to manipulate, transform and manage natural language data in arbitrary ways.

The most interesting feature of computer natural language research to many workers, however, is that the methods a computer uses in comprehending language may serve as a model in the attempt to study human language comprehension, or language comprehension in general. In order to understand the methodology which computer scientists have adopted in the study area, it is necessary to examine the research work which has so far been produced. In this chapter, we will first make a brief survey of current work in computational

linguistics which is related to our research in section I-2. Then in section I-3 we will explain the outline of this thesis, which describes results obtained during the past few years by the author.

I-2 Language Understanding System --- A Survey of Related Work

Initially the major problem of computational linguistics was machine translation. Machine translation (MT) programs were designed to accept a text in one language (*source-language*) as input and produce as output a text in another language (*object-language* or *target-language*) that has the same meaning. The initial approach to MT was very simple-minded. Giant dictionaries were prepared in which the equivalent target-language word was assigned to each word of the source-language. For each word in an input text, an equivalent in the target-language was found. Then, rules were applied that in order to transform the word ordering of the source language into that of the target-language. However, the results obtained by this approach were not encouraging. Researchers in MT came to believe that a more precise understanding of the syntax of natural languages was necessary for better translation. The first attempt was to apply a certain set of syntactic rules to the input sentences and to obtain certain intermediate structures which represent syntactic constructions of sentences. These intermediate structures were usually called *parse trees*. A certain set of rules was to be applied on a parse tree to obtain the corresponding syntactic construction of the target-language. Generation of a sentence in the target-language, then, was defined as the mapping of the syntactic structure into lexical strings. In this respect, Chomsky's theories began to gain favor. In *Syntactic Structure*, Chomsky(1957) outlined the theory of transformational grammar. This was a new

syntactic theory, and people working on machine translation considered the programming of transformational grammars as the solution to the MT problem. However, it did not take long for the researchers to feel that this approach had come to a deadlock. It appeared that even a simple sentence such as

Time flies like an arrow

has many possible syntactic structures (Kuno, 1963). In order to select a single correct syntactic structure from among others, it seemed necessary to grapple with the problem of *semantics* of natural language. Moreover, it seemed that not only semantics but also knowledge of the physical world and inferential abilities based upon it were needed for translation. Y. Wilks (1975a,b) presents an example as follows, which requires such kinds of abilities. Suppose that an input text is

The soldiers fired at the women and I saw several of them fall.

His problem was to translate the sentence to French. In analyzing the above sentence, the question arises of whether *them* refers to *soldiers* or *women* because the choice will result in a differently gendered pronoun in French. In order to disambiguate the reference of such an anaphoric expression, the system will have to be able to infer that things fired at often fall or at least are much more likely to fall than things doing the firing. Hence there must be access to inferencial information here, above and beyond the meanings of the constituent words.

What is really needed for good translation is a successful computational model of language comprehension. Such computational models should include the components, for not only syntactic but also semantic and pragmatic analyses, and the model should also be able to explain how these components interact with each other in the process of comprehension. Such computational models must be more

precise and performance-oriented than those proposed by linguists or psychologists.

T. Winograd stated in his thesis (Winograd, 1972) his attitude to computational linguistics as follows :

> When a human reader sees a sentence, he uses knowledge to understand it. This includes not only grammar, but also his knowledge about words, the context of the sentence, and most important, his knowledge about the subject matter. A computer program supplied with only a grammar for manipulating the syntax of language could not produce a translation of reasonable quality.

and therefore,

> If we really want computers to understand us, we need to give them ability to use more knowledge. In addition to a grammar of the language, they need to have all sorts of knowledge about the subject they are discussing, and they have to use reasoning to combine facts in the right way to understand a sentence and respond to it.

These attitudes have come to be common among the researchers in computational linguistics. In recent years, the research focus in computational linguistics gradually shifted from the development of techniques for language processing to the construction of computational models of understanding natural language. The researchers were first convinced that the extent to which a system understands input would be demonstrated by the behaviour of the system when a certain sentence, a question or a request, was received. They thought : if a system could make conversations with human as if it were a human being, it could be claimed that the system understood natural language. Remarkable advances have been made in this direction. The programs which *understand* English by R.F.Simmons (1975), L.S.Coles (1968), B.Raphael (1968), T.Winograd (1972), W.A.Woods (1972) and others are remarkable examples. In particular, SHRDLU by Winograd and LUNAR by Woods showed remarkable performance in conversations about the 'block world' and 'moon rocks', respectively. Because these systems can give *meaningful* replies to certain ques-

tions or perform *appropriate* actions for given requests, the extent to which the system understands the input is reflected more directly in the performance of the system than in other systems such as those for machine translation. However, because different conversation systems treat different domains of discourse and different domains usually require different mechanisms of inference and frameworks for representing knowledge, it is very difficult to claim that a certain system is better in general than other systems. For example, it is difficult to say whether SHRDLU is better than LUNAR or vice versa. The conversational program ELIZA by J.Weizenbaum (1966) could simulate the conversational mode of a Rogerian psychotherapist as examplified in the following brief excerpt, without any *understanding* mechanisms (Responses from the computer are in the upper case letters).

Well, my boyfriend made me come here.
YOUR BOYFRIEND MADE YOU COME HERE?
He says I'm depressed much of the time.
I AM SORRY TO HEAR YOU ARE DEPRESSED.
It's true, I am unhappy.
DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY?
Etc.

Though ELIZA showed a remarkable and very natural mode of conversation, its internal mechanism was far from a reasonable model of human comprehension. However, if one is limited to superficial observations of relationships between human inputs and system's responses, one can hardly say that the recently developed conversation systems simulate human conversational behaviour definitely better than ELIZA. Here we encounter, the difficult problem of how to evaluate or compare the performances obtained by different systems. One must carefully examine in detail the nature of the mechanisms and representational frameworks are employed, what internal processings were responsible for producing responses, and to what

extent the internal processing simulates the human behaviour of understanding language.

The following criteria should be useful for the evaluation and comparison of various understanding systems :

1. *Concrete vs. Abstract* : As described earlier, natural language processings can be divided into several components such as morphological, syntactic, semantic and pragmatic components of analysis. These components are theoretically organized in a hierarchical form, typically constructed as shown in Fig. I-1.

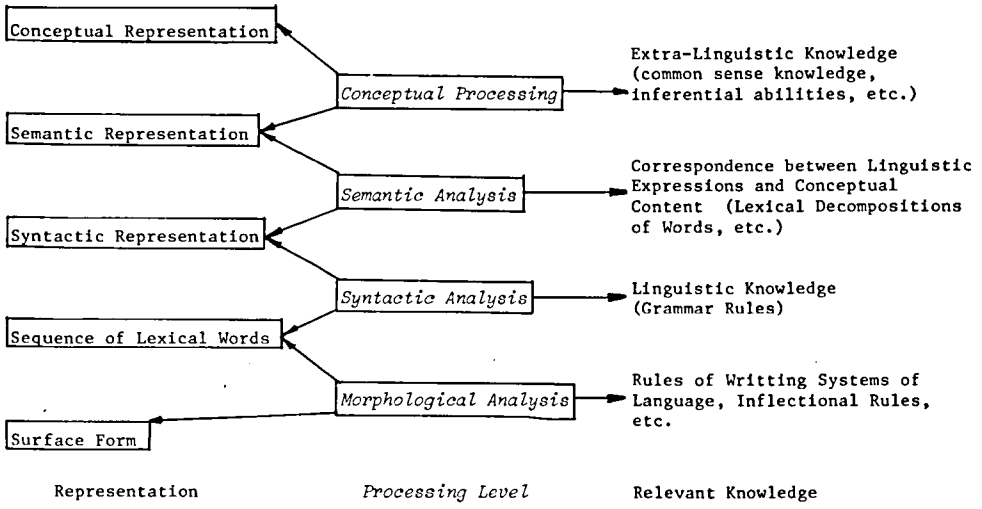


Fig. I-1 Concrete vs. Abstract Hierarchy

The lower component level in this hierarchy is, the more dependent the processing is on the peculiarities of the language being processed. Because the characteristics of data and algorithms employed in the morphological analysis are highly dependent on the writing systems of languages, the morphological procedure for Japanese may be quite different from that for English. On the other hand, the structures which represent conceptual meaning of a sentence and the

inference procedures which will be applied on the conceptual structures may well proye to be independent of the actual language. Because the understanding of a sentence is reflected in the form of internal representation, a useful criterion for the evaluation of understanding systems is the degree to which the representation becomes independent of the surface structure of a sentence. In other words, it is desired that the two sentences which have different surface constructions but convey the same meaning for a human be converted to the same internal representation. When this criterion is employed it can be safely claimed that ELIZA did not understand the input sentences because ELIZA did not use an internal representation of any kind.

The surface forms of the sentences

Jim bought a car from John.
John sold a car to Jim.

are completely different. The syntactic parse trees for these sentences may also be quite different. Therefore, through the observation of surface forms or parse trees one cannot say anything about the relationships of the above two sentences. The method for representation of meaning developed by R.F.Simmons (1973) may explain the equivalence of the meanings of these two sentences. The representation of meaning in his system was based on the notion of *cases* developed by linguists such as C.J.Fillomre (1968), M.Celce (1972) and so on. A set of transformation rules on case structures was provided in his system to explain the equivalence of meaning of two different sentences. It can be claimed that his representation is more abstract than ordinary parse trees. In another system for representing meaning, Schank and others (1973) proposed a certain set of conceptual primitives. Using these conceptual primitives, their system MARGIE represented the meanings of the above two sentences in the same structure. Therefore, no transformation rules were necessary to explain the equivalence. In this sense, Schank's

representation based on *conceptual dependency theory* is more abstract or conceptual than Simmon's representation based on cases. Of course, the linguistic analysis component in the Schank's system is responsible to transform input sentences into such abstract structures, and may become more complicated than Simmon's.

2. *Local vs. Global* : Early language processing systems such as those for machine translation treated each sentence as a separate processing unit. Syntactic analysis in such systems was applied to each sentence at a time. The result of the analysis was usually a parse tree. However, the designers of recent understanding systems believe that a more global analysis and representation framework is needed for better understanding. The anaphoric expression given above in the *soldiers and women* example is a good example which requires inter-sentential processings. The system SHRDLU by Winograd demonstrated that, in order to resolve the ambiguities of anaphoric references, a system must be able not only to represent the meaning of each sentence but also to maintain a continuity of representation regarding the situations it learns about. The discussions by E.Charniak (1972) developed this idea further. He considered the following example.

Janet needed some money. She got her piggybank(PB) and started to shake it. Finally some money came out.

To claim that a system understands the above story, it should be able to answer correctly the following questions, which are very easy for humans.

Why did Janet get the PB ?
Did Janet get the money ?
Why was the PB shaken ?

In order to answer these questions, the system should have a great deal of common sense knowledge which is not explicitly mentioned in the story, and should utilize it in order to understand

sentences in a definite context. The descriptions based on semantic or conceptual primitives mentioned aboye are chiefly concerned with the meaning of a single word or sentence. The descriptions can be seen to still be linguistic ones(One can easily find many similarities between the descriptions based on Schank's conceptual dependency theory and the descriptions developed by the linguists in the school of generative semantics). However, the description needed in computer models of language comprehension is more related to pragmatics and less to linguistics. That is, the description should include what events usually follow a certain event, for what purpose human performs a certain action, and so on.

Recently M.Minsky (1974), Schank (1975a,b), Charniak (1977), Wilks (1977) and others have proposed data-structures which represent human common-sense knowledge about more global and stereotyped situations. These data-structures have been called Frames, Scripts, Pseudo-Texts and so on. SAM (1975b), by Schank et.al.,Ms.Maloprop (1977) by Charniak and other recently developed systems which employ such kinds of data-structure, therefore, can be said to have the ability of understanding texts more globally than MARGIE, SHRDLU, LUNAR etc.

In the following section we will discuss the approach we take towards those two questions of representation, i.e. that of local information vs.global information, and that of representation in terms of concrete, linguistic features vs. more abstract,conceptual features.

I-3 Outline of the Thesis

The ultimate goal of our research is to construct a system which can reply appropriately in natural language sentences to ques-

tions posed to it also in natural language. At the present we have chosen the field of elementary chemistry as the *micro-world* of our system. The scope of questions which our system is expected to handle includes not only simple questions such as

'Do you know the molecular weight of hydrochlorine ?'

but also somewhat difficult questions such as shown in Fig. I-2.

Human : There exists about 10 cm^3 of certain liquid in a test tube. The pH value of the liquid is about 3. When I put 1 gr of zinc in it, the liquid dissolved the metal, and some hydrogen was created.
Can you imazine what the liquid is ?

Computer : Because the pH value of the liquid is less than 7, it is acid. The acid which dissolves zinc is either nitric acid, sulfuric acid or hydrochloric acid .
But I can not determine which one the liquid is.

Human : In order to distinguish those three kinds of chemical material, what shall I do ?

.....

Human : We have 10 cc of aqueous solution in a beaker. The solute is supposed to be sodium, and the density is about 10 %. Can you calculate the mol-density ?

Computer : I can't, but if either the total mass or the consistency is known, I can calculate it.

.....

Fig. I-2 Hypothetical Dialogue with the System

To accomplish this goal there are many difficult problems which must be solved beforehand. First of all we must be able to represent knowledge about the environments of chemical experiments,

and also be able to transform given questions into internal structures on which a problem solver and/or deduction program can work effectively. Natural language sentences usually contain much irrelevant information for such tasks and necessary information is often expressed only in very implicit forms or sometimes not at all. Second, we must develop an appropriate framework in which we can represent technical knowledge which is specific to elementary chemistry and procedures which utilize this knowledge to solve problems and make deductions. In the chemical field there are many types of knowledge, that is, numerical data (molecular weight, ...), reaction relationships (Sulfuric acid reacts copper and produces hydrogen, ...), and other, more complex forms of knowledge (If a liquid dissolves copper and the pH of the liquid is less than 4, then the liquid is either sulfuric acid, nitric acid, or hydrochloric acid, ...). We must integrate these fragments of knowledge into a framework and develop procedures which can utilize them efficiently during the problem solving process.

In the following pages, we will describe these problems in more detail and discuss our attitude to them.

We consider the process of understanding natural language as the combined activities of extracting relevant information from sentences, and of representing the extracted information in structures which are convenient for the procedures that perform some expected tasks. Our task is to construct a system which can answer questions in chemical fields. Therefore, the final structure which should be extracted from sentences is the structure which the problem solving procedure will work on. We will describe this structure in Chapter V. The structure has a somewhat rigid form though also a clear logical meaning. However, it is our contention that it is very difficult to apply such logically rigid knowledge directly to the analysis of sentences. It is best to provide in the system several intermediate levels of representation and the procedures which utilize them. It

is also our contention that the two issues mentioned in section I-2, namely concrete ys. abstract and local ys. global, are somewhat independent, and the confusion between these should be carefully avoided. In recent research on language understanding systems it is often the case that these two issues are confused. Thus knowledge for global processing was represented at a highly conceptual level. Schank's Scripts, Charniak's Frames and so on are examples. However, we believe that some of global processing can be performed much easier by utilizing intermediate levels of representation and that sometimes we can not adequately deal with a certain phenomena of natural language without recourse to representations at very low levels. We do not claim that processing at the conceptual level is not important, but that there are many useful clues in the lower levels to control conceptual level processing. Because natural language is the only medium by which humans can communicate their more complex ideas, it is obviously true that the natural language sentences contain many useful keys for guiding the interpretations of the sentences. The reason of the failures of the early syntax-based language processing systems is not, of course, that they used syntax, but that they relied only on syntactic processing and thus grasped syntactic structures in wrong ways. It is necessary to properly appreciate the role of syntax and even the role of the surface forms of sentences. We will explain this by a simple example(the meanings of the symbols in these examples will be given in pp65 - 68).

Though we claimed in Section 1-2 that the sentences

(1) John-WA KURUMA-O Jim-NI UTTA.
SUBJ *a car* OBJ IOBJ *to sell*(past tense)

John sold a car to Jim.

(2) Jim-WA John-KARA KURUMA-O KATTA.
SUBJ FROM *a car* OBJ *to buy*(past tense)

Jim bought a car from John.

have the same meaning, it is always difficult to judge whether two sentences have the same meanings. This depends on what one means by the word *meaning*. From the computational point of view, we provide

certain internal representations for processing them. If two sentences have different import and thus different influences on certain inferential processes, it is necessary that they be expressed by different internal representations. Consider the following simple sequences of sentence.

(3) John-WA KURUMA-O Jim-NI UTTA.
SUBJ *a car* OBJ IOBJ *to sell* (past tense)

John sold a car to Jim.

SOSHITE SONO-KANE-O Mary-NI HARATTA.
and *the money* OBJ IOBJ *to pay* (past tense)

And (someone) paid the money to Mary.

(4) Jim-WA John-KARA KURUMA-O KATTA.
SUBJ FROM *a car* OBJ *to buy* (past tense)

Jim bought a car from John.

SOSHITE SONO-KANE-O Mary-NI HARATTA.
and *the money* OBJ IOBJ *to pay* (past tense)

And (someone) paid the money to Mary.

The first lines of (3) and (4) are just the sentences (1) and (2) respectively, and the second lines of (3) and (4) are identical. In the second sentences of the texts, the specification of the person who paid the money to Mary is omitted. We must identify from the first sentence the person who paid the money. If we generate the same internal representations for the sentences (1) and (2), the analysis of the second sentences in both cases will proceed in completely the same manner, and therefore the same person would be identified with the person who paid the money. However, in the case of (3), the ordinary Japanese reader would definitely consider John as the person who paid the money while in the case of (4) they would identify Jim as the payer. These distinctions can be based solely on the observations of the surface forms of the sentences (1) and (2). The relevant observations are:

The Japanese postposition '-WA' indicates the focus of attention and the focussed word or phrase is often omitted in succeeding

sentences.

In Japanese, identical case elements in succeeding sentences are apt to be omitted. The AGENT of the first sentence in (3) is John so that John is considered to be the most feasible candidate for the filler of the omitted case element in the second sentence while the AGENT of the first sentence in(4) is Jim.

These observations are sufficient to determine who paid the money. However, systems which are solely based on the conceptual level of representations such as MARGIE could not understand the above sequences of sentences properly. Systems based on logical deductions may also fail to process the sentences. Such a system might reason as follows:

> If a person buys something from somebody, he should pay money to somebody.
>
> Jim bought a car from John.
>
> Therefore, Jim should pay the money to John.
>
> However, Jim paid the money to Mary.
>
> Therefore, the sequence of the events in these texts should be invalid. I can not understand what happened at all.

Wilks' system, based on the notion of *Preference Semantics*, might behave much better than those just mentioned. He claims that the approach to the understanding language through artificial intelligence places too much emphasis on problem solving and logical deduction facilities. Furthermore, he feels that more intuitive modes of reasoning are necessary to understand ordinary texts. However, his system does not pay any attention to the syntactic structures or surface forms of sentences. His system uses syntactic clues only for the segmentation of sentences. Therefore, his system might identify in the both cases the person who paid the money as being Jim, by thinking roughly as follows.

If a person buys something from somebody, he probably pays

money for it. At least, it is more appropriate to consider that the person who bought something as paying money than the person who sold an article. Therefore, it is more semantically feasible that Jim paid the money than that John did.

Because the human process of understanding natural language may involve many different levels of processing, and each processing level seems to require a different kind of representation, we should provide all these levels in a language understanding system. The systems developed so far were much biased towards performing the final tasks of the systems. That is, they often used a single internal representation at every stage of language processing. This single representation was chosen to be convenient for performing final tasks such as question-answering in particularly restricted domains, paraphrasing of sentences and so on. This is also the case in more recently proposed systems which are equipped with data structures for capturing the more global characteristics of natural language texts. Schank's scripts, for example, consists of several script statements, each of which is assumed to be expressed by his conceptual dependency formalism. It seems that there is an implicit hypothesis that global understanding can be obtained by only working with the conceptual level of representations. However, as the above example shows, this hypothesis is not always true. Syntactic structures and the surface forms of sentences in some cases play an important roles in understanding the global structure of texts. At any rate, because these examples provide useful keys to guide higher levels of processing, we should always keep them in mind .

The above example offers another interesting problem. In the second sentence we find the anaphoric expression SONO-KANE(*the money*), though money is not explicitely mentioned in the first sentence. A human reader can immediately understand that it refers to the money which Jim paid John for the car. This type of reasoning, of course, is based on human empirical knowledge and therefore cannot be

explained within the framework of syntax. We need certain representations at another level, and certain procedures which utilize them. At this level of representation, it is desired that the sentences (1) and (2) be transformed into the same structure, because the surface verbs KAU(*to buy*) and URU(*to sell*) denote the same event in the actual world and to a good approximation have the same implications at more abstract levels of consideration.

In our system, the following levels of representation are provided at present. (see Fig. I-3)

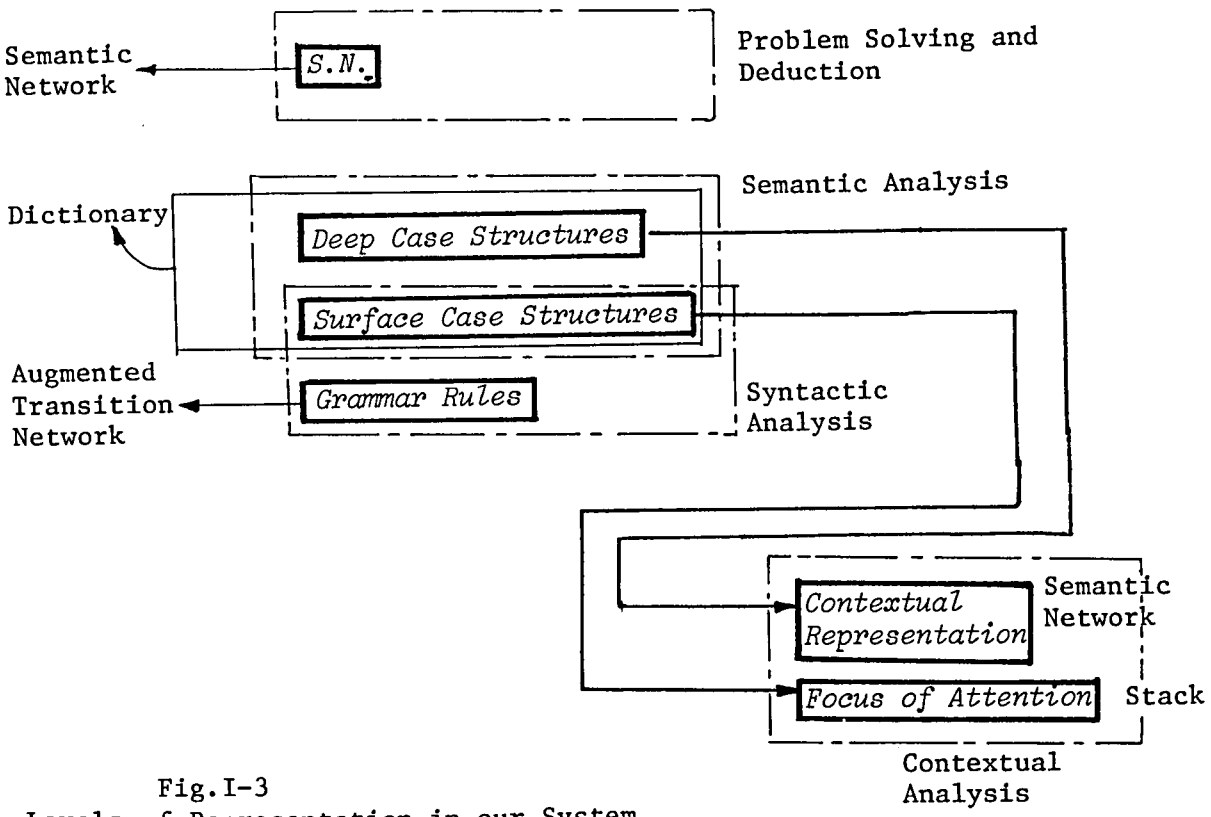


Fig.I-3
Levels of Representation in our System

1. *Surface Case Structures* (SCS) : We feel that the notion of case proposed by linguists such as C.J.Fillmore (1968) and M.Celce (1973) are useful for bridging the gaps between syntactic

structures and the semantics of sentences. However, the grammars which they proposed were constructed for the purpose of generating sentences from *deep case structures*. These methods, then, must be carefully reconsidered before they can be applied to the problem of sentence recognition. As we pointed out earlier, sentences which have the same meaning, that is, which can be generated from the same deep case structures, may have different imports in global processing. This is natural because the speaker's intent is usually reflected in the transformation processes from the deep case structures to the surface forms, and this intent plays a central role in understanding global characteristics of the texts.

We provided our system with two different levels of representation based on cases. One is called surface case structure and the other deep case structure. The surface forms of sentences are almost directly reflected in the SCS.

The purpose of the representation at this level is as follows :

(1) In English, the role (or case) of a noun phrase in a sentence is usually evident in the surface ordering of phrases. In Japanese, on the other hand, case is usually marked by the postposition attached to the phrase. The ordering of phrases is relatively free. Therefore, it may not be feasible to apply such structures as *semantic templates* (Wilks) directly to the surface forms of sentences. Our SCS can be seen as a form of semantic template augmented with surface case markers. Because it is rather easy to translate Japanese postpositions into case markers, it is easier to check whether a sentence matches a certain SCS than to attempt to match a DCS directly to the sentence. Without this level of representation, very complicated analyses (including perhaps inverse applications of transformational rules) may be required to fit a DCS to the surface form of a sentence.

(2) This representation level contains many useful clues

for disambiguating certain types of anaphoric expressions and for processing omitted words or phrases. Such useful information would be lost at the level of the DCS. The SCS, however, preserves them in such a way that they can be extracted much more easily than from the surface form.

An important consideration in choosing the form of internal representation is the ease with which the procedure working on the representations can manipulate the structure and how easily relevant information can be retrieved from the structure. We have found that the case-frame-like structures shown in Fig. I-4 are convenient for integrating the semantics of words in a sentence with the syntactic structure of a sentence, and that the case-frame-like structures are more convenient for the later utilization than the parse trees in the conventional forms such as shown in Fig. I-5.

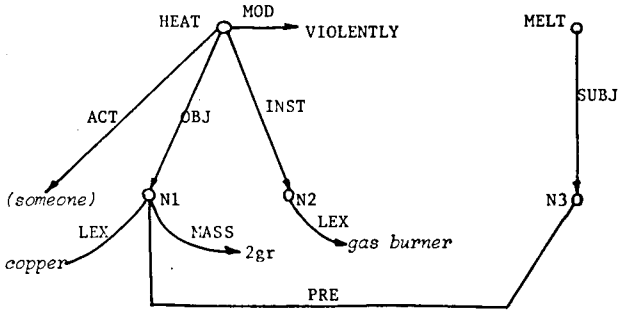


Fig. I-4 Case Representation

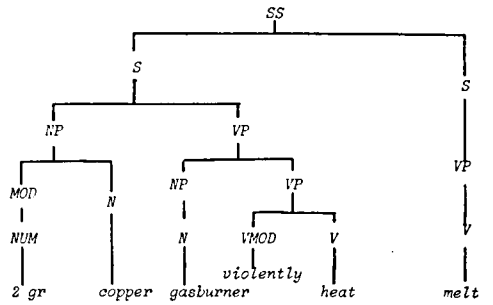


Fig. I-5 Conventional Parse Tree
(The meaning of the symbols in this figure will be given in pp 65 - 68).

Each verb has its own usage patterns (possibly more than one). These usage patterns are tabulated in the Verb Dictionary as the surface case frames of the verb. One of these case frames is to be instantiated by an actual sentence. The description for a noun at this level is also constructed in a fashion similar to the case frame of a verb. Some types of nouns also have several relational slots in their descriptions which are to be filled by other words or phrases. We have classified the nouns which appear in the chemical field into five groups and prepared different frameworks of expressing their meanings.

2. *Deep Case Structures* (DCS) : In human natural language understanding, the ability to perform intuitive reasoning plays a central role. Such ability is based on forms of human empirical knowledge which do not always satisfy the logical criteria of truth. Associative functions relating to semantic similarities between words, semantic depth of an interpretation and probability of associative occurrence of events are inherent factors in the process of intuitive understanding and reasoning. Because a certain event in the actual world can be expressed by several different surface forms, there may be several different SCS's for an event. This is also the case for the representation of noun phrases in SCS level. We need another level of representations in which we can express human empirical knowledge naturally. For this purpose, we supplied the system with a framework of representations called Deep Case Structures (DCS). This representation level roughly corresponds to the deep case structures of Fillmore. His case system was developed in order to explain various kinds of language phenomena. Therefore, he proposed a set of primitive cases (Agentive, Instrumental, Dative, Factive, Locative and Objective) aimed at being sufficient to deal with linguistic phenomena. We are, however, interested in the notion of deep cases for representing various kinds of human empirical knowledge. For this purpose, we must reconsider Fillmore's set of

primitive cases carefully and revise them, if necessary.

Each SCS in the Verb Dictionary has a corresponding DCS. By utilizing this description, an SCS which is instantiated by an actual sentence may be transformed into its corresponding DCS. Different SCS's which refer to the same event or concept are transformed into a same structure at this level. Moreover, additional information is attached to a DCS, such as, what events will be likely to succeed the current event, what changes are expected among the participants of the event (usually expressed by noun phrases which are related to certain deep cases of the verb and so on). Utilizing this additional information it is possible to do various types of global processing.

3. *Contextual Representations by a Semantic Network :* When one cannot understand a sentence satisfactorily, one refers back to the preceding sentences to obtain a key to its understanding. If one cannot find what is needed, one leaves the question pending and proceeds to the next sentence. This procedure is necessary in order to understand linguistic structures which are longer than a single sentence.

Both the SCS and DCS are stored in the dictionary, and in this sense are permanent knowledge. On the other hand, in order for the system to be able to perform certain kinds of contextual processing, it is neccessary to supply both contextual representations and procedures which utilize these representations. We use a semantic network for this purpose. The network is constructed from the DCS's of input sentences. In order to reflect information contained in SCS's and the various associative expectations clustered around each DCS, the network is augmented by three types of lists. These lists contain information about what objects are currently the focus of attention, what objects are expected to appear, what events are expected to occur next and so on. Various heuristics are used to arrange the ordering of elements in these lists. These heuristics

utilize clues from the SCS's and from knowledge attached to the DCS's of input sentences. The procedures for contextual processings access the semantic netowork through these lists.

4. *Logical Representation in the Semantic Network S.N.* : The information attached to a DCS includes vague and empirical forms of knowledge which are not always logically valid. However, in order to permit the system to answer many questions a human might ask, we should supply some procedure which performs problem solving and/or logical deductions. Though the DCS representation and the semantic network for contextual knowledge are rich enough for the purpose of analysis of sentences, they contain much irrelevant information and sometimes lack necessary information for problem solving and logical deductions. For example, as pointed out by L.Schubert (1976), case relationships do not have any meaningful logical implications, though they are important in the analysis of sentences. We have provided another type of semantic network called S.N. for the representation of knowledge which is specific to the field of chemistry. The S.N. shares various features in common with recently developed network systems by G.Hendrix (1975a,b), S.Scrag (1976), J.Sowa (1976), L.Schubert (1976), J.Mylopoulos (1975,1977), J.Minker (1977), R.F.Simmons (1977) and so on. These are intended to clarify the expressive power of conventional semantic networks, by exploiting their logical properties. Their basic methods are based on the expression of certain logical constructs within network representations. However, perhaps too much emphasis has been placed on the direct correspondence between the syntax of network representations and the syntax of ordinary logical formulas. As a result, their representations have come to be basically graphical notations for the corresponding logical formulas, though some useful ideas were developed, for example, the partitioning of networks by Hendrix, introduction of Lambda abstraction in networks by Schubert, and so on. An advantage of network representation is, we believe, that

both the selection of relevant knowledge and the execution of logical operations may be carried out at the same time by simply traversing networks. We have developed S.N. to preserve this advantage while preserving the logical validity of both representations and operations. Moreover, the semantic network we developed provides a framework in which various kinds of knowledge, such as procedural knowledge, knowledge about external data bases and so on, can be naturally embedded. We have also developed a programming language suitable for describing and constructing semantic networks.

In this thesis, we will describe in detail the organization of each component of our question-answering system, which has been developed in accordance with our position on the above issues. The outline of this thesis is as follows.

As described in section I-2, a language analysis program consists of many components, for instance,morphological, syntactic, semantic and pragmatic analyses. It is a difficult problem to control and invoke these subcomponents at appropriate times during the analysis. We have developed a new programming language called PLATON(Programming LAnguage for Tree OperatioN) for this purpose. The language is developed to simplify the writing natural language analysis programs. Based on the model of ATN(Augmented Transition Network) proposed by W. Woods (1970), PLATON is provided with various additional facilities such as automatic backtracking, pattern matching and so on. The pattern matching process has the facility to extract substructures from an input sentence and invoke appropriate semantic and contextual checking functions. Using this language we can easily obtain natural and flexible interactions between syntactic and other components. A detailed description of PLATON will be given in Chapter II.

Chapter III and Chapter IV are devoted to a detailed explanation of the program which analyze Japanese sentences. Chapter III is mainly concerned with the description of word meanings (using the

SCS and DCS) in the dictionary and with procedures for analyzing noun phrases and simple sentences. The problem of analyzing a long noun phrase in Japanese has never been attacked prior to this research. It is a very difficult problem because a long noun phrase usually contains very few syntactic clues to its analysis. We will show that the relationships among the elements of a noun phrase can be appropriately determined by utilizing various types semantic constraints. These semantic constraints are expressed in case-frame-like structures in the dictionary. In Chapter IV, we will explain how contextual information may be represented by a semantic network and how it may be utilized to disambiguate reference in anaphoric expressions, or to determine omitted words or phrases. Some experimental results will also be given in this chapter.

In Chapter V, we will describe in detail the construction of the semantic network S.N., in which we can express specific knowledge about chemistry. Because our question-answering system is at present restricted to elementary chemistry, the examples of representations given in this chapter are all taken from that field. S.N., however is a generalframework for representing knowledge in problem solving and logical deductions. The problem solver which utilizes these representations also has wide applicability, which permits the expression of knowledge relating to many different fields. Designers of language understanding systems often bypass the problem of morphological analysis simply by claiming that words in texts directly correspond to lexical entries. This is unrealistic for Japanese, because we have no definite word delimiters in Japanese (such as spaces in English), and because compounding is much more frequently used in Japanese than in English. Some morphological processing must be applied to texts before syntactic and semantic analysis. The morphological analysis is highly dependent on characteristics of the language itself, especially on the writing system. Since the writing system of Japanese is quite different from those of European languages,

we have had to devise our own morphological analysis procedure for Japanese. The detailed construction of the morphological analysis procedure for Japanese and some experimental results are given in Chapter VI. A large computerized Japanese dictionary (about 70,000 lexical entries) is used for morphological analysis. The data structure and some efficient access methods for this dictionary are also discussed.

In Chapter VII, we summarize the researches in this thesis and makes some brief comments on the future problems in this research area.

26 項欠

CHAPTER II

A PROGRAMMING LANGUAGE FOR NATURAL LANGUAGE ANALYSIS

II-1 Introduction

PLATON(Programming LAnguage for Tree OperatioN), which has the facilities of pattern matching and flexible backtracking, is described. The language is developed to simplify writing natural language analysis programs. The pattern matching process has the facility to extract sub-strings from the input sentence and invoke semantic and contextual checking functions. This makes the interactions between syntactic and other components very easy. The designers of natural language analysis programs have not to pay any attention to the interactions. We think that a backtracking mechanism is also necessary for language understanding as in other fields of artificial intelligence. We can set up arbitrary numbers of decision points in the analysis programs. If processing results in a failure, a message which expresses the cause of the failure will be sent up. The control will be modified appropriately according to the message. This prevents 'wasted backtrackings' during the analysis process of sentences, and enables us to write fairly complicated non-deterministic programs in a simple manner. In this chapter we describe the specifications and control mechanisms of PLATON. An example of structural analysis using PLATON is also described.

II-2 Outline of PLATON

There are two key issues in analyzing natural language by

computer;

1) *how to represent knowledge (semantics, pragmatics and the state of the world - context* , and 2) *how to advance the programming technology appropriate for syntactic-semantic, syntactic-contextual interface.*

The point in designing a programming language is to make this kind of programming less painful.

Traditional systems which represent grammars as a set of rewriting rules usually have poor control mechanisms, and flexible interaction between the syntactic and other components is not possible. Systems in which rules of grammars are embedded in procedures, on the other hand, make it possible to intermix the syntactic and semantic analyses in an intimate way. However, these systems are apt to destroy the intelligibility and regularity of natural language grammars, because in these systems both rules and their control mechanisms are contained in the same program.

Recently various systems for natural language analysis have been developed. T. Winograd's (1972) "PROGRAMMAR" is a typical example of procedure oriented systems. In this system the syntactic and other components can interact closely in the course of analyzing sentences. However, details of the program are lost in the richness of this interaction. LINGOL, developed by V. Pratt (1973, 1975) at MIT, is a language appropriate to syntax-semantics interface and in which it is easy to write grammars in the form of rewriting rules. The TAUM group at Montreal University (1971) has evolved a programming language named System-Q in which expressions of trees, strings and lists of them can be matched against partial expressions (structural patterns) containing variables and can be transformed in an arbitrary fashion.

In Japan, H. TANAKA et. al, (1977) developed Extented LINGOL at ETL. The Extended LINGOL was a modified version of LINGOL and

equipped with additional facilities which were not provided with the original LINGOL, such as specifying rules of the morphological analysis of Japanese and so on.

The augmented transition network (ATN) proposed by W. Woods (1970), from our point of view, gives an especially good framework for natural language analysis systems. One of the most attractive features is the clear discrimination between grammatical rules and the control mechanism. This enables us to develop the model by adding various facilities to its control mechanism.

The ATN model has the following additional merits:

1. It provides power of expression equivalent to transformational grammars.

2. It maintains much of the readability of context-free grammars.

3. Rules of a grammar can be changed easily, so we can improve them through a trial-and-error process while writing the grammar.

4. It is possible to impose various types of semantic and pragmatic conditions on the branches between states. By doing this, close interactions between the syntactic and other components can easily be accomplished.

However ATN has the following shortcomings, especially when we apply it to the parsing of Japanese sentences:

1. It scans words one-by-one from the leftmost end of an input sentence, checks the applicability of a rule, and makes the transition from one state to another. This method may be well suited to English sentences, but because the order of words and phrases in Japanese sentences is relatively free, it is preferable to check the applicability of a rule by a flexible pattern-matching method. In addition, without a pattern-matching mechanism, a single rewriting rule of an ordinary grammar is often to be expressed by several rules belonging to different states in Woods' ATN parser.

2. An ATN model essentially performs a kind of top-down analysis of sentences. Therefore recovery from failures in prediction is most difficult.

Considering these factors, we developed PLATON (a Programming LAnguage for Tree-OperatioN), which is based on the ATN model and has various additional capabilities such as pattern-matching, flexible backtracking, and so on. As in System-Q and LINGOL, PLATON's pattern-matching facility makes it easy to write rewriting rules. Moreover, it extracts substructures from the inputs and invokes appropriate semantic and contextual checking functions. These may be arbitrary LISP functions defined by the user, the arguments of which are the extracted substructures.

A backtracking mechanism is also necessary for language understanding as in other fields of artificial intelligence. During the analysis, various sorts of heuristic information should be utilizable. At any stage, analysis based on criteria which may relate to syntactic, semantic or contextual considerations taken separately may be unreliable. An interpretation which fulfils all the criteria, however, may be the correct one. The program should be designed such that it can choose the most satisfactory rule from many candidates according to the criteria at hand. In further processing, if the choice is found to be wrong by other criteria, the program must be able to backtrack to the point at which the relevant decision was made. In PLATON we can easily set up arbitrary numbers of decision points in the program. Then, if subsequent processing results in some failure, control will come back to the points relevant to the cause of the failure.

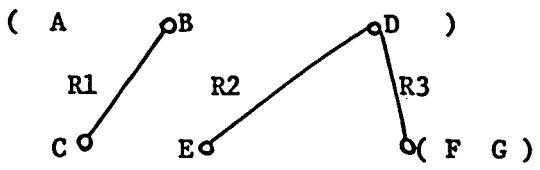
II-3 Pattern-Matching in PLATON

Before proceeding to the detailed description of PLATON, we will explain the representation schema for input sentences and parsed trees. The process of analyzing a sentence, roughly speaking, may be regarded as the process of transforming an ordered list of words to a tree structure, which shows explicitly the interrelationships of each word in the input sentence. During the process, trees which correspond to the parts already analyzed, and lists which have not been processed yet, may coexist together in a single structure. We therefore wish to represent such a coexisting structure of trees and lists. A list structure is a structure in which the order of elements is not changeable. On the other hand, a tree structure consists of a single root node and several nodes which are tied to the root node by distinguishable relations. Because relations between the root and the other nodes are explicitly specified, the order of nodes in a tree is changeable except for the root node which is placed in the leftmost position. Different matching schemas will be applied to trees and lists.

The formal definition of such coexisting structures is as follows. <structure> is the fundamental data-structure into which all data processed by PLATON must be transformed. Hereafter we refer to this as the "structure". The formal definition of <structure> is:

```
<structure> ;:=<tree>|<list>
<list>  ::= (*<structures>)
<structures>  ;:=|<structure><structures>
<tree>  ;:=<node>|(<node><branches>)
<branches>  ;:=<branch>|<branch><branches>
<branch>  ::= (<relation><tree>)
<node>  ::=<list>|ARBITRARY LISP-ATOM
<relation>  ::= ARBITRARY LISP-ATOM
            (* in the definition of  list  is a terminal symbol.)
```

A simple example is shown in Fig. II-1.



Coexisting Structure of Trees and Lists

```
( * A (B (R1 C)) (D (R2 E)
        (R3 ( * F G))) )
```

Corresponding Expression in PLATON

Fig. II-1 Expression of Structure in PLATON

Two lists which have the same elements but different orderings (for example, (*A B C) and (*A C B)), should be regarded as different structures. On the other hand, two tree structures such as (A (R1 B) (R2 C)) and (A (R2 C) (R1 B)) are regarded as identical. Besides the usual rewrite rules which treat such strings, structural patterns which contain variable expressions are permitted in PLATON. The PLATON-interpreter matches structural patterns containing variable expressions against the structure under process and checks whether the specified pattern is found in it. At the same time, the variables in the pattern are bound to the corresponding substructures.

Variables in patterns are indicated as :X (X is an arbitrary LISP atom). The following can be expressed by variables in the above definition of <structure> :

(1) arbitrary numbers of <structures>, that is to say, list elements in the definition if <list> (Fig. II-2, Ex. 1). We can also specify the number of list elements by indicating variables as :X+number. For example, the variable :D2 will match with two elements in a list.

(2) arbitrary numbers of <branches>, in the definition of <tree> (Fig. II-2, Ex. 2).

(3) <tree> in the definition of <branch> (Fig. II-2, Ex. 3).

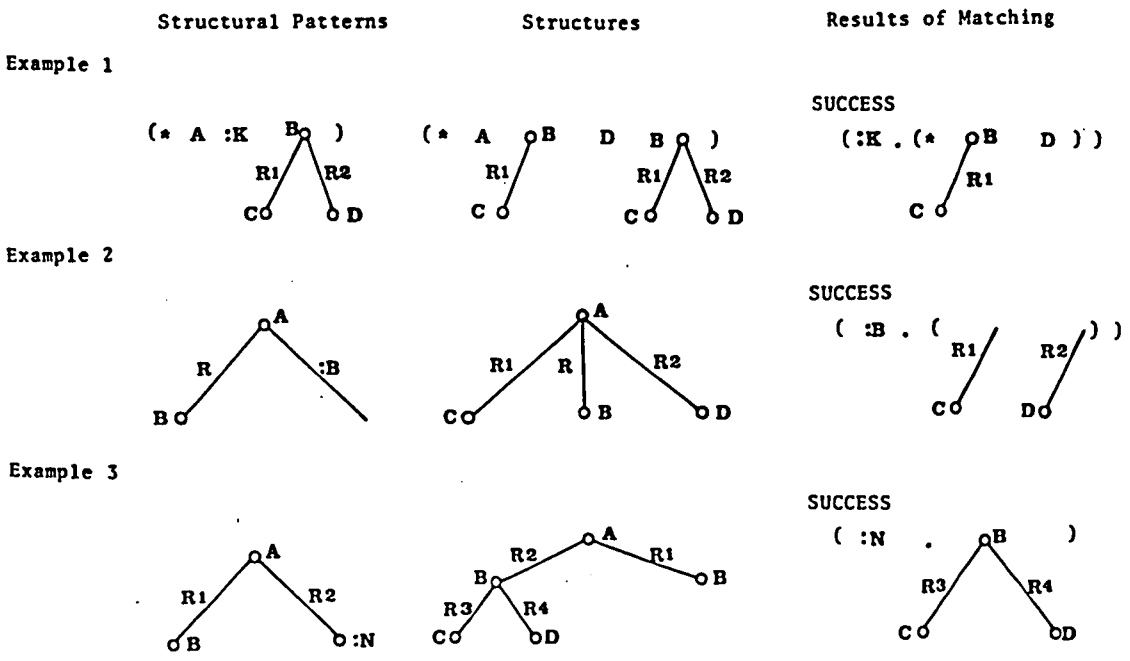


Fig. II-2 Illustration of Matching

We shall call such structural patterns <structure-1>. By using the same variable several times in a pattern, we can express a structure in which the same sub-structure appears in two or more different places.

II-4 Representation of Grammar in PLATON

A grammar, whether generative or analytical, is represented as a directed graph with labeled states and branches. There is one state distinguished as the Start State and a set of states called Final States. Each branch is a rewriting rule and has the following elements:

(1) *applicability conditions of the rule, typically represented as a structural pattern*

(2) *actions which must be executed, if the rule is applicable*

(3) *a structural pattern into which the input structure should be transformed*

Each state has several branches ordered according to the preference of the rules. When the control jumps to a state, it checks the rules associated with the state one-by-one until it finds an applicable rule. If such a rule is found, the input structure is transformed into another structure specified by the rule and the control makes the state transition.

In addition to the above basic mechanism the system is provided with push-down and pop-up operations. The push-down operation is such that in the process of applying a rule, several substructures are extracted from the whole structure by variable binding mechanisms of pattern-matching. Then each is analyzed from a different state. The pop-up operation is such that after each substructure is analyzed appropriately, control comes back to the suspended rule and execution continues. Using these operations, embedded structures

The detailed explanation of push-down and pop-up operations will be given in section II-5.

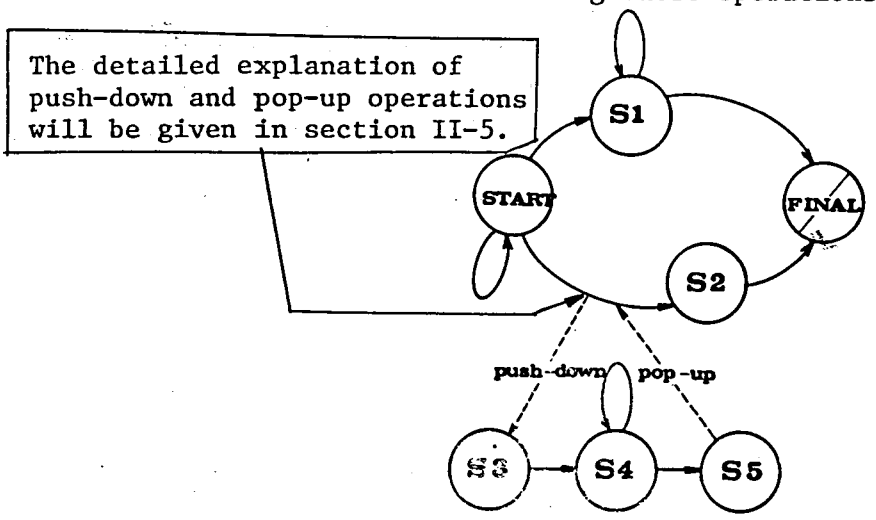


Fig. II-3 State Diagram

can be handled easily (See Fig. II-3).

Table 1 shows the formal definition of a grammar of PLATON. It is shown that branches or rewriting rules in an ATN parser correspond to six-tuples (i.e., <pcon> , <strx> , <con> , (<trans>), (<acts>), <end>). <strx> corresponds to the left side of a rewriting rule and describes the structural pattern to which a rule is applicable. <strx> is, by definition

(1) / or

(2) <structure-1>

/ shows that a rule is applicable no matter what the structure under process is. The variables used in <structure-1> are bound to corresponding substructures when matching succeeds. The results of Example 1 (See Fig. II-2) indicate that the variable :K is bound to the substructure (* (B (R1 C)) D).

The scope of variable binding is limited to within the realm of the particular rule. The same variable name in different rules has different interpretations. In this sense, :X-type variables in <structure-1> are called Local Variables. On the other hand, we can store certain kinds of results from the application of rules in registers and refer back to them in different rules. These are variables which we call registers. They are represented by the symbols /X (X is an arbitrary LISP atom).

Besides the pattern-matching, <pcon> and <con> can also check the applicability of a rule. Certain parts of the results from the application of previous rules are contained in registers, not in the structure. We can check the contents of these registers by using <pcon> -part functions like GR, GU, etc. (these functions are listed in Table. II-2) and other LISP functions defined by the usual LISP function, DEFINE. (See following page for Table. II-2.)

Semantic and contextual co-ordination between substructures can be checked by using appropriate functions in the <con>-part of a rule. Semantic and contextual analyses cannot be expressed in the

```
<grammar>  :: = ( <states> )

<states>   :: = <state> | <state> <states>

<state>    :: = ( <state-name>  <rules> )

<rules>    :: = |<rule> <rules>

<rule>     :: = ( <pcon> <strx> <con> ( <trans> )( <acts> ) <end> )

<trans>    :: = | <transit> <trans>

<transit>  :: = (( <state-name> <structure-2> { <register-name> | <variable-name> } ) <errorps>)

<errorps>  :: = | <errorp> <errorps>

<errorp>   :: = ( <failure-message> <act> <pros> )

<pros>     :: = <pro> | <pro> <pros>

<pro>      :: = (EXEC <trans> ) |(TRANS ( <state-name> <stry> ))

<end>      :: = (NEXT <state-name> <stry> )
                |(NEXTB <state-name> <stry> )
                |(POP <stry> ) | (FM <failure-message> )

<acts>     :: = | <act> <acts>

<act>      :: = <form> | (SR <register-name> <form> )
                | (SU <register-name> <form> )
                |(SD <register-name> <form> )

<strx>     :: = <structure-1> | /

<stry>     :: = <structure-2> | /

<pcon>,<con>:: = <form>

<form>     :: = (GR <register-name> ) | (GV <variable- name> )
              |(TR <structure-2> ) |(TR /) | ARBITRARY LISP FORM

<variable-  :: = :X (X is an arbitrary LISP atom)
 name>

<register-  :: = /X (X is an arbitrary LISP atom)
 name>
```

Table. II-1 Formal Definition of Grammar in PLATON

Function	Argument	Effect	Value
SR	<register-name> LISP - <form>	SR stores the result of the evaluation of the 2nd argument in the register.	the result of the evaluation of the 2nd argument
SV	<variable-name> LISP - <form>	SV stores the result of the evaluation of the 2nd argument in the variable	the result of the evaluation of the 2nd argument
GR	<register-name>	GR get the content of the register	the content of the register
GV	<variable-name>	GV gets the value of the variable	the value of the variable
TR	<structure-2> or /	TR transforms the variables and registers in the structural pattern into their values.	the transformed structure
SU	<register-name> LISP - <form>	SU sets the reigster of the higher level processing.	the result of the evaluation of the 2nd argument
SD	<register-name> LISP - <form>	SD sets the register of the lower level processing.	the result of the evaluation of the 2nd argument
GU	<register-name>	GU gets the content of the register of the higher level.	the content of the register
PUSHR	<register-name> LISP - <form>	PUSHR is defined as the following. (SR <register-name> (CONS <form> (GR <register-name>)))	the result of the evaluation of the 2nd argument

Table. II-2 Functions of PLATON

form of simple rewriting rule. These analyses have differing requirements such as lexical information about words which may in turn represent knowledge of the world and contextual information which may express the state of the world. We can use arbitrary LISP-forms in the <con>-part, depending on the semantic and contextual models we choose. For example, suppose

```
strx = (* (ADJ ( TOK :N ))( N(TOK :N1 ))  :I)
con = ( SEM :N :N1 )
```

Here TOK is the link between a word and its part of speech. :N and :N1 are the words of an input sentence. SEM is a function defined by the user which checks the semantic co-ordination between the adjective :N and the noun :N1. By this function SEM, we can search, if necessary, through both lexical entries and the contextual data bases.

With this approach, if a certain syntactic pattern is found in the input structure, it is possible for an appropriate semantic function to be called. Hence the intimate interactions between syntactic and semantic components can be obtained easily without destroying the clarity of natural language grammars.

Arbitrary LISP-forms can be also used in <act>-portion. They will be evaluated when the rule is applied. If necessary, we can set intermediate results into registers and variables by using the functions listed in Table. II-2.

<end> comprises four varieties, and rules are divided into four types according to their <end> types.

1. NEXT-type: The <end> is in the form (NEXT <state-name> <stry>). The <stry> corresponds to the right side of a rewriting rule, and represents the transformed structure. A rule of this type causes state-transition to the <state-name>, when it is applied.
2. NEXTB-type: This rule also causes state-transition. Unlike with the NEXT-type, state-saving is done and if further processing results in some failures, control comes back to the state

where this rule is applied. The environments, that is, the contents of various registers will be restored, and the next rule belonging to this state will be tried.

3. POP-type: The <end> -part of this type is in the form (POP <stry>). When it is applied, the processing of this level is ended and the control returns to the higher level with the value <stry>.
4. FM-type: The <end> -part of this type is in the form (FM <failure-message>). The side effects of the processing at this level, that is, register settings and so on, are cancelled (see section II-4).

In <stry> we can use two kinds of variables, that is, the variables used in <strx> and registers. We find this structural pattern, called <structure-2>, more suitable for writing transformational rules than Woods' BUILDQ-operation. By way of illustration consider the following:

```
input string        = (* C D E ( A ( R1 (* B ))) F G )
strx                = (* :I ( A ( R1 :N )) :J )
stry                = (* (A (R1 (* :I :N ))( R2 /REG )) :J)
the content of /REG = (G (R3 H ))
```

As the result of matching, the variables :I, :N and :J are bound to the substructures (* C D E), (* B) and (* F G) respectively. The result of evaluating the <stry> is

(* (A (R1 (* C D E B))(R2 (G (R3 H)))) F G).

If the rule is a POP-type one, then this structure will be returned to the higher level processing. If it is NEXT- or NEXTB-type, then the control will move to the specified state with this structure.

II-5 Push-down and Pop-up Operations --- Error Recoveries

By means of NEXTB-type rules, we can set up decision points in a program. We can also do this by using *push-down* and *pop-up* operations. A rule in PLATON finds particular syntactic clues by its structural description <strx>, and at the same time, extracts substructures from the input string. From the structural description it is predicted that the substructures may have particular constructions, that is, they may comprise noun phrases, relative clauses or whatever. It is necessary to transfer the substructures to states appropriate for analyzing these constructions predicted and to return the analyzed structures back into the appropriate places. In PLATON, these operations can be described in the <trans>-part of a rule. For example, suppose the <trans> -part of a rule is

(((S1 :K :K)) ((S2 (* :I :J)/REG)))

When the control interprets this statement, the substructures corresponding to the variable :K and (* :I :J) are transferred to the states S1 and S2 respectively. If the processings starting from these states are normally completed (by a POP-type rule), then the results are stored in the variable :K and the register /REG. In this manner, by means of the push-down and pop-up mechanisms, substructures can be analyzed from appropriate states. Processing from these states, however, may sometimes result in failure. That is, predictions that certain relationships will be found among the elements of substructures may not be fulfilled. In such instances the pushed down state will send an error-message appropriate to the cause of the failure by an FM-type rule. An FM-type rule points out that a certain error has occurred in the processing. If NEXTB-type rules were used in the previous processing at this level, control will go back to the most recently used NEXTB-type rule. If NEXTB-type rules were not used at this processing level, the error-message specified by the FM-type

rule will be sent to the <trans> -part of the rule which directed this push-down operation (see Fig. II-4).

According to these error-messages, control-flow can be changed appropriately. For example, we can direct processing by describing the <trans> -part in the following way.

```
( (( S1 :K :K )( ERR1 (EXEC (( S5 :K :K )) (( S6 (* :I :J ) /REG ))))
                 ( ERR2 (TRANS ( S8 /)))
  (( S2 (* :I :J ) /REG )) )
```

In the above example, the processing of the substructure :K from the state S1 will produce one of the following three results. According to the returned value, the appropriate step will be taken:

(1) *Normal return*: the processing of :K is ended by a POP-type rule. The result is stored in the variable :K and the next push-down performed, that is (* :I :J) will be transferred to the state S2.

(2) *Return with an error-message*: the processing of :K results in a failure and an FM-type rule sends up an error-message. If the message is ERR1, then :K and (* :I :J) will be analyzed from the states S5 and S6 respectively (EXEC-type). If it is ERR2, the interpreter will give up the application of the present rule, and pass the control to another state S8 (TRANS-type). If it is neither ERR1 nor ERR2, the same step as (3) will be taken.

(3) *Return with the value NIL*: the processing from the state S1 will send up the value NIL if it runs into a blind alley, that is, there are no applicable rules. The interpreter will give up the application of the present rule and proceed to the next rule attached to this state.

Mechanisms which enable control flow to be appropriately changed according to the error-messages from lower level processings are not found in Woods' ATN parser. We can obtain flexible back-tracking facilities by combining these mechanisms with NEXTB-type

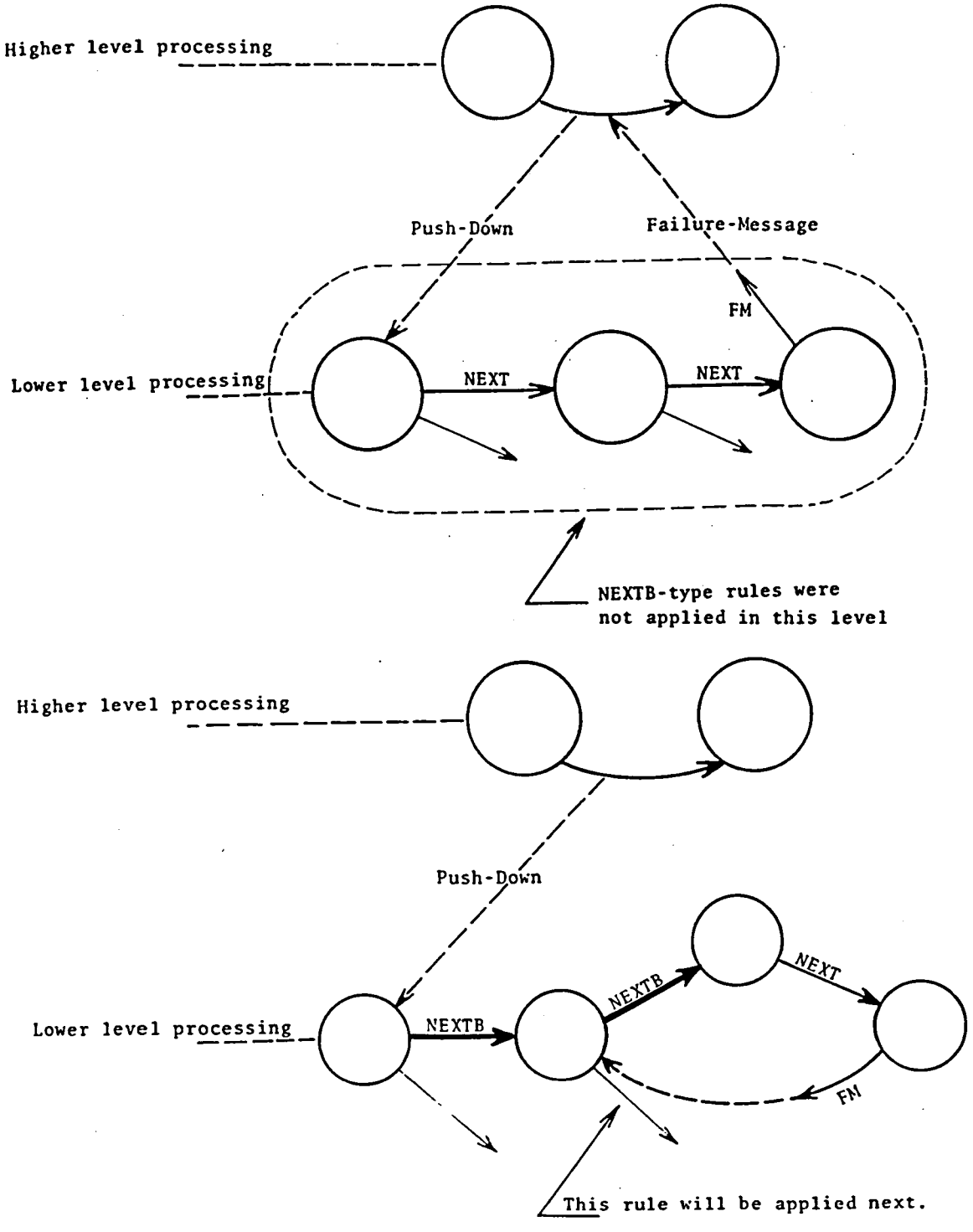


Fig. II-4 Illustration of Backtracking

rules.

II-6 A Simple Example

By using PLATON, we have developed an analysis program for Japanese sentences. The detailed construction of the program and the examples which show how the various facilities of PLATON can be used in writing grammars will be given in Chapter III and Chapter IV. In this chapter, we illustrate by a rather simple example how structural analysis is performed by PLATON rules, and how the facility of backtracking is used in the analysis process.

Japanese is a typical example of an SOV-language in which the object and other constituents governed by a verb usually appear before the verb in a sentence. A typical construction of a Japanese sentence is shown in Fig. II-5.

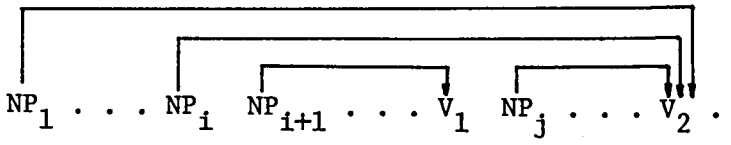


Fig.II-5 Typical Construction of a Japanese Sentence

A verb may govern several noun phrases preceding it. A relative clause modifying a noun may appear in the form '-- verb + noun --'. The right boundary of the clause is easily identified by finding the verb. The left boundary is often much more difficult to identify. In Fig.II-5, the noun phrase NP_{i+1} is a case element of the verb V_1. On the other hand, the noun phrase NP_i is governed by the verb V_2. Because the rule of projections holds in Japanese as in other languages, all the noun phrases between NP_{i+1} and V_1 are governed by V_1,

and the noun phrases before NP_i are governed by V_2. However, in the course of analysis, such boundaries cannot be determined uniquely. The analysis program fixes a temporary boundary and proceeds to the next step in processing. If the temporary boundary is not correct, the succeeding processing will fail and the control will come back to the point at which the temporary boundary was fixed.

Now we will show a simple example of structural analysis by PLATON. The example explains how the backtracking facility is used in analyzing Japanese sentences. Because we want to visualize the operations of PLATON without bothering with microscopic details of Japanese sentences, we will take an imaginary problem as an example.

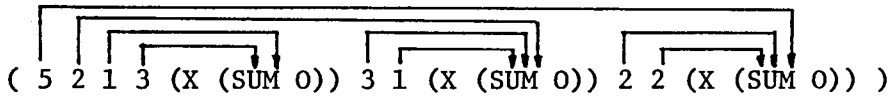
An input string is assumed to be a list. The elements of the list are integers and trees of the form (X (SUM 0)). Here 'X' may be regarded as a term modified by 'SUM 0'. These two kinds of elements are arranged in an arbitrary order, except that the last element is the tree (X(SUM 0)). The following is an example of an input string:

(* 5 2 1 3 (X (SUM 0)) 3 1 (X (SUM 0)) 2 2 (X (SUM 0)))

The result of the transformation is expected to be in the following form:

(* (X (SUM 4)) (X SUM 6)) (X (SUM 9)))

This result is regarded as representing the following relationships between integers and 'X'.



The number associated with an 'X' by the relation 'SUM' shows the sum of the integers which are governed by the 'X'. We can look upon the relations between integers and an 'X' as the relations between noun phrases and the verb in Japanese sentences. The result of the

analysis is assumed to satisfy the following conditions.

(1) Governor-governed relationships between integers and an 'X' must obey the projection rule (i.e., clauses do not overlap).

(2) As a simulation of a semantic restriction, we attach a condition that the sum of the integers governed by an 'X' should not exceed ten.

(3) As a simulation of a contextual restriction, we attach the condition that a result (* (X (SUM num-1)) (X (SUM num-2)) . . . . (X (SUM num-N))) should maintain the relation, num-1 $\leqq$ num-2 $\leqq$ . . . . $\leqq$ num-N.

A set of rules is shown in the following. The corresponding state-diagram is shown in Fig. II-6.

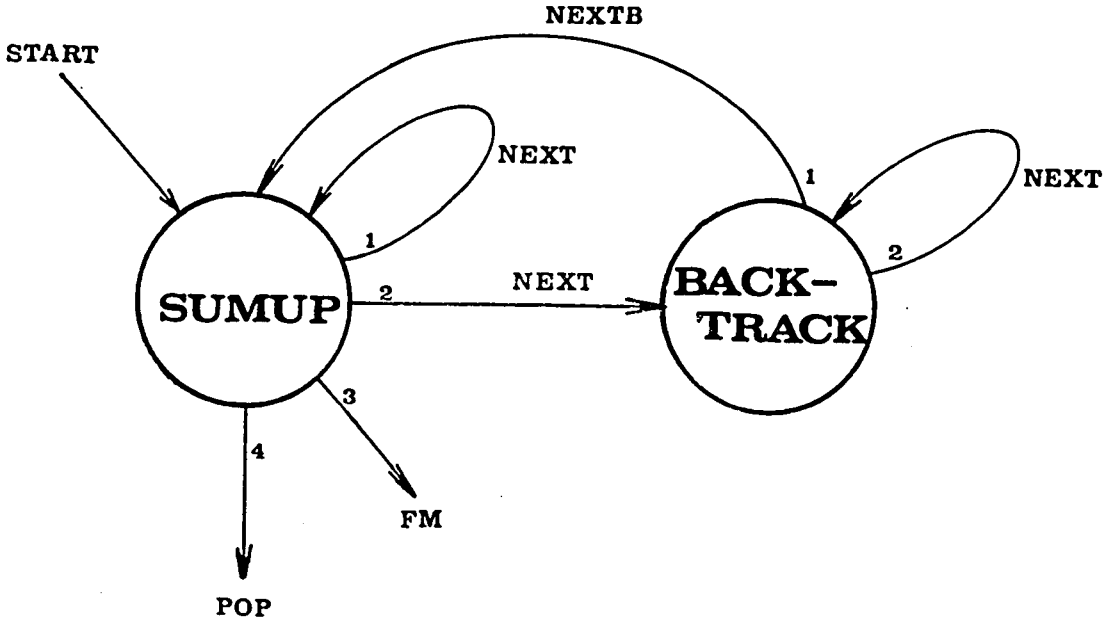


Fig. II-6 State Diagram of a Simple Example

```
SUMUP -1-  strx: = (* :I :I1 (X (SUM :N)) :J)
           con:  = ( GREATERP 10 (PLUS :N :I1))
           act:  = ( (SV :N (PLUS :N :I1 ))
                     (PUSHR  /REG :I1) )
           end:  = (NEXT SUMUP (* :I (X (SUM :N)) :J) )

      -2-  strx: = (* :I (X (SUM :N)) :J)
           con:  = (CONTEXTCHECK /RESULT (TR (X (SUM :N))))
           act:  = NIL
           end:  = (NEXT BACKTRACK /)

      -3-  strx: = (* :I (X (SUM :N)) :J)
           con:  = T
           act:  = NIL
           end:  = (FM ERROR)

      -4-  strx: = (*)
           con:  = T
           act:  = ( (SR /RESULT (CONS 'X /RESULT )) )
           end:  = (POP /RESULT)

BACKTRACK

      -1-  strx: = (* :I (X (SUM :N)) :J)
           con:  = T
           act:=( (SR /REG NIL)
                  (SR /RESULT (APPEND /RESULT ( TR (X (SUM :N))))) )
           end:  = (NEXTB SUMUP (* :I :J ))

      -2-  strx: = (* :I (X (SUM :N)) :J)
           con:  = T
           act:  = ( (POPR /TEMP /REG)
                     (SV :N (MINUS :N /TEMP)) )
           end:  = (NEXT BACKTRACK (* :I /TEMP (X (SUM :N   :J) )
```

The input string is the list shown in Fig.II-6. Since the start state is SUMUP, the first rule attached to this state is applied. This rule will find the leftmost 'X' and an integer just before the 'X' (by SUMUP -1-, strx). The variable :I1 is bound to this integer. This integer is added to the sum of the integers, :N, if the total does not exceed ten (SUMUP -1-, con). PUSHR, used in the <act>-part, is a PLATON function which puts the second argument on the head of the first argument which is the register /REG(SUMUP -1-, act). After this rule is applied, the control will enter the state SUMUP again (SUMUP -1-, end). That is, this rule is applied until there

are no integers before the first 'X' or the sum of the integers exceeds ten. As the result, the environment is the following:

structure under processing
= (* 5 (X (SUM 6)) 3 1 (X (SUM 0)) 2 2 (X (SUM 0)))
relationship temporarily fixed between integers and 'X'
= (5 2 1 3 X 3 1 X 2 2 X)
content of /REG
= (2 1 3)

The second rule of SUMUP will be applied next. This rule checks by its <con> -part whether the result at hand satisfies the third condition, that is, the contextual restriction. Because the content of /RESULT is NIL, the function CONTEXTCHECK returns the value T (SUMUP -2-, con). So this rule is applicable. Control makes the state-transition to the state BACKTRACK (SUMUP -2-, end). Because the first rule of BACKTRACK is a NEXTB-type rule, state-saving is performed. That is, the following environment is saved:

content of /REG = (2 1 3)
content of / RESULT = NIL
structure under processing =
(* 5 (X (SUM 6)) 3 1 (X (SUM 0)) 2 2 (X (SUM 0)))

By this rule, the registers /REG and /RESULT are set as follows (BACKTRACK -1-, act).

/REG : = NIL
/RESULT: = ((X (SUM 6)))

And the structure is transformed to

(* 5 3 1 (X (SUM 0)) 2 2 (X (SUM 0))).

A NEXTB-type rule causes a state transition as does a NEXT-type rule. So control returns to the state SUMUP (BACKTRACK -1-, end). At this state, a process similar to the one described above is performed. As a result, the following governor-governed relationships are

established.

Here the bold lines indicate the newly established relationships. By the first rule of BACKTRACK the following environment is saved.

content of /REG = (5 3 1)
content of /RESULT = ((X (SUM 6)))
structure under processing = (* (X (SUM 9)) 2 2 (X (SUM 0)))

And /REG and /RESULT are set as the following (BACKTRACK -1-. act).

/REG. = NIL
/RESULT = ((X (SUM 6)) (X (SUM 9)))

The transformed structure is (BACKTRACK -1-, end)

(* 2 2 (X (SUM 0)))

The control is transferred to the state SUMUP. By applying the first rule of this state repeatedly on the above structure the following structure is obtained.

(* (X SUM 4)))

However, this result does not satisfy the contextual restriction. So the application of the second rule of SUMUP fails because the function CONTEXTCHECK used in <con>-part returns the value NIL (SUMUP -2-, con). That is:

contextcheck [((X (SUM 6))(X (SUM 9))) : (X (SUM 4))] = NIL

The third rule, therefore, will be applied next. Because this rule is an FM-type rule(SUMUP -3-, end), it causes an error and control comes back to the point at which a NEXTB-type rule was applied most recently. The saved environment is restored. This is:

/REG: = (5 3 1)
/RESULT: = ((X (SUM 6)))

sructure under processing: = (* (X (SUM 9)) 2 2 (X (SUM 0)))

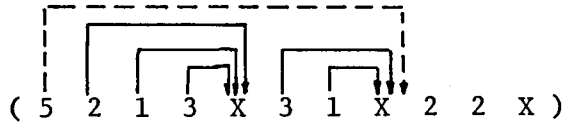
Then by applying the second rule of BACKTRACK, the governor-governed relationship established lastly in the previous process is cancelled. The structure and the register /REG are changed as below (BACKTRACK -2-, act):

/REG: = (3 1)
structure under processing: = (* 5 (X (SUM 4)) 2 2 (X (SUM 0)))

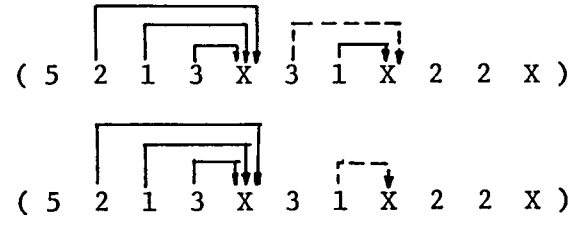
Control enters the BACKTRACK state again. The application of the first rule saves the environment:

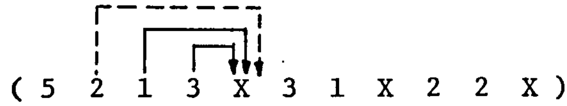
content of /REG = (3 1)
content of /RESULT = ((X (SUM 6))
structure under processing = (* 5 (X (SUM 4)) 2 2 (X (SUM 0)))

That is, the relationship indicated by the dotted line in the following is cancelled:

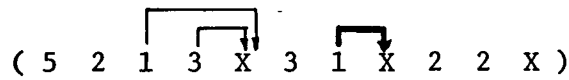


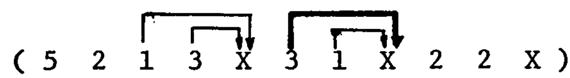
Control transfers to the state SUMUP (BACKTRACK -1-, end) and a similar process is performed. However, because the governor-governed relationship between the integer 5 and the second 'X' is cancelled, the sum of the integers governed by the first 'X', (2 1 3), is greater than that of the second 'X', (3 1). The contextual condition, therefore, is not fulfilled, and the application of the second rule of SUMUP will not succeed. So the temporarily established relationships will be cancelled one-by-one as follows.

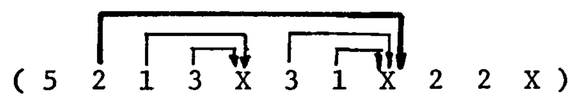


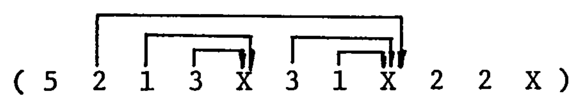


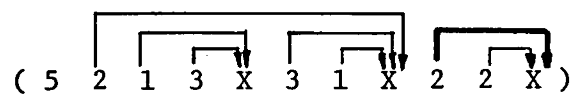
After these relationships have been cancelled, the desired result is obtained by the following sequence.

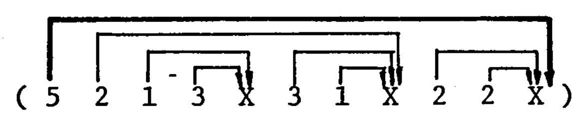












At the final stage of the processing, the fourth rule of SUMUP, a POP-type rule, is applied and returns the value

(* (X (SUM 4)) (X (SUM 6)) (X (SUM 9))).

II-7 Conclusion

We have described in this chapter a programming language called PLATON for natural language processing. The language has several additional capabilities beyond the ATN parser of W. Woods.

Grammars written in the language not only maintain clarity of representation but also provide adequately a natural interface between the syntactic component and other components. By means of the pattern-matching facility, we can write grammars in a quite natural manner. And because of the PLATON variable binding mechanism, semantic and contextual LISP functions are easily incorporated in syntactic patterns.

Flexible backtracking mechanisms and push-down operations make complicated non-deterministic processing possible in a very simple way.

We have developed an analysis program for Japanese using this language. The program can accept fairly complicated sentences in a textbook of elementary chemistry. It can utilize the lexical and contextual information of chemistry adequately during the analysis. We will explain the detailed construction of the analysis program in the following chapters (Chapter III and Chapter IV).

Perhaps, PLATON itself must be equipped with more semantics and context-oriented operations such as specified lexical descriptions and functions using them. However, which description method is most efficient, and moreover, what semantic information must be stored in the lexicon, are not yet entirely clear. So, as the first step, PLATON leaves many parts of these problems for the user to specify by LISP programs.

52項欠

CHAPTER III

DESCRIPTION OF MEANING AND SEMANTIC ANALYSIS OF JAPANESE SENTENCES

III-1 Introduction

In this chapter and the following chapter (Chapter IV) we describe the organization of the natural language parser. This forms an important part of our question-answering system. The parser can transform fairly complex sentences into abstract structures marked for case. It utilizes detailed semantic dictionary descriptions and contextual information abstracted from the preceding sentences. Some intuitively appealing schemas of representation for the semantic descriptions of words are discussed. Meanings of verbs are described by using *case* concept. Additional information is attached to *case frames* of each verb to indicate what changes the case elements in the frame may undergo and what events may occur in succession. Meanings of nouns are also expressed in case-frame-like descriptions. Nouns also have relational slots which must be filled in by other words or phrases.

Several new techniques based on heuristically admissible operations are presented in this chapter to analyze: 1) *complex and long noun phrases* 2) *conjunctive phrases* and 3) *a simple sentence*.

For the present, we have confined the domain of the system to the field of elementary chemistry where we can describe the semantic world in rather concrete terms. At the same time, various complex

events occur in this field. For example, substances which participate in particular events may disappear, new substances may emerge, or some properties of the substances may be altered. To treat these complex situations, it is necessary to formally represent relationships between events and changes of state and to devise an appropriate schema for representing context.

In most approaches to the understanding of natural language through artificial intelligence, schemas which entail rigid logical operations are used to represent both knowledge and context. Logical operations appear to be necessary for solving some kinds of problems in natural language, especially at the deep deductive level of understanding. However, intuitive reasoning is not easily formalized in terms of logical operations. It is our contention that intuitive reasoning is completely based on the language activities in the human brain. Associative functions relating to semantic similarities between words, semantic depth of an interpretation and probability of associative occurrence of events are inherent factors in intuitive understanding and the reasoning process.

Y. Wilks (1975) in his system carries out intuitive reasoning by employing the notion of *semantic preference*. His system seems to work well on analyzing local relationships among words. However, in order to analyze more global relationships (e.g., in dealing with complex cases of anaphora) we require access to more information than can be contained in formulas (templates) associated with the lexicon. We find Wilks' use of *CS-inference rules* rather awkward. The system would be much improved if accompanied by an appropriate schema for representing context.

Case grammar sentence-analysis theories such as those of C.J. Fillmore (1968) and M.Celce et.al.(1972) are based on the semantic relationships between verbs and nouns -- events and concepts. R. F. Simmons (1973; 1975), D. A. Norman (1973), D. E. Rumelhart (1973) and so on follow these theories to represent knowledge and

context in their systems. We also adopted case grammar and modified it to account for Japanese sentences. We represent context in the form of a semantic network. An input sentence is transformed into a corresponding *deep case structure*. This structure is assimilated with the semantic network constructed from previous sentences.

Japanese is a typical SOV language. The word order is rather arbitrary except that the main verb comes last. Cases such as subjective, objective and dative are syntactically indicated by postpositions, but a postposition can be used for several deep cases ambiguously. Hence the determination of underlying sentential structures rests heavily on an understanding of the semantic relations between the main verb and nouns. Moreover in Japanese the words which are essential in understanding a sentence are often omitted without pronominal reflexes. Our system can infer from the semantic descriptions of words what kinds of phrases should be supplied to fill lexical gaps and search the contextual representation to find appropriate fillers.

The final analysis produced by our parser is a semantic network. This is to be used for the internal representation of data in a question-answering system or as an intermediate expression in machine translation. We will transform this semantic network into more logically rigid structure called S. N.. The construction of the S. N. and problem solving procedures based on it will be explained in Chapter V.

The parser consists essentially of four fixed components:

1) *The grammar consists of rules written in PLATON.* A grammatical rule in PLATON consists of two parts: pattern rewrite which is expressed as a pair of syntactic patterns, and semantic and contextual check which is an arbitrary LISP function. When a rule is to be applied, the semantic and contextual check is employed to determine whether the rule is semantically and contextually feasible. For the present we have about two hundred rules for the analysis of

Japanese sentences. These rules are devised to combine various syntactic patterns in Japanese with appropriate semantic and contextual checking functions.

2) *In the dictionary are stored words along with their various semantic relationships.* We express the meaning of a word in terms of how it may be related to other words. The meaning of a verb is described in the form of case frames in the verb dictionary. Two different levels of case frame descriptions are prepared for a verb. One is called SCS (*Surface Case Structure*) and the other is called DCS (*Deep Case Structure*). An SCS corresponds to a usage pattern in the surface level of the verb. The diversity of the meanings of Japanese postpositions are resolved by this description. On the other hand, a DCS describes an activity pattern of the action denoted by the verb. The DCS roughly corresponds to the deep case structures of linguistics. The DCS describes which case relations the activity entails and what kind of referents will be appropriate for each case slot. Additional information is provided with a DCS, which feeds into the *change* or *causative* component used by D.A.Norman (1973). Such information indicates how one activity pattern may be related to another by causal relationships and what related change may occur in the semantic network representing context. From such information we can infer what activities and changes will follow the present activity.

The meanings of nouns are also expressed in the case-frame-like descriptions. They also have relational slots which will be filled in by other words or phrases. The formats of meaning descriptions for a noun and a verb are given in section III-2.

3) *Contextual representation in a Semantic Network.* The contextual representation is similar in form to the semantic network of R. F. Simmons (1973; 1975) or the node space of Norman (1973) In this representation there are two kinds of nodes. The C-node corresponds to a concept typically expressed by a noun. The S-node

corresponds to an event. An event is a realization of an action pattern and each argument of the pattern is assigned a C-node. C-nodes are related to S-nodes by the case-labeled relations. These relations are bidirectional.

The following list shows the relations used in the network:

(i) *Deep Case Relations*: ACT, OBJ, PLACE, TIME -- A deep case relation connects an S-node with its argument C-node.

(ii) *Attributive Relations*: VOLUME , COLOR, MASS, SHAPE -- An attribute relation connects a C-node with its value. We can distinguish two C-nodes associated with the same lexical entry but different values of attributes.

(iii) *Token substitution*: TOK -- TOK is used to connect a node with a lexical entry.

(iv) *Event-Event Relation*: CAUSE, IMPLY -- Two S-nodes are sometimes connected by a particular relation. The relations are sometimes expressed explicitly in the surface sentence by a special conjunction such as NODE (because), NARABA (if) and so on.

In our system the semantic network is accompanied by special lists (*Noun Stack*-NS, *Hypothetical Noun Stack*-HNS, *Trapping List*-TL). We call these lists Intermediate Term Memory. Contextual functions work on these lists to search appropriate nodes of the semantic network which correspond to the referents of anaphoric expressions or the unexpressed elements of sentences. These are described in Chapter IV.

(4) *Semantic and Contextual functions are programmed in LISP*. These functions are incorporated in the PLATON rules along with rewriting patterns. A contextual function takes as arguments the semantic constraints a target node must satisfy and returns the appropriate node if it is found in the semantic network. A semantic function checks descriptions in the dictionary to determine whether the combination of two words is semantically permissible. For analyzing noun-noun combinations, we provide sixteen semantic functions.

III-2 LEXICAL DESCRIPTIONS OF WORDS

III-2-1 Noun Description

Most nouns have a definite meaning by themselves. We call these *Entity Nouns*. An entity noun is considered to represent a set of objects, and therefore is taken as a name of the set. The objects belonging to the set may share the same properties. By introducing another property the set may be divided into a number of subsets, each of which is expressed by another noun. We describe such set-inclusion relationships and set properties in the noun dictionary.

We represent a property of a noun by an attribute-value pair expressed as (A V). For instance, the dictionary entries for the nouns 'material' and 'liquid' are :

material : ((SP)(ATTR(STATE)(MASS)(COLOR)(SHAPE) ---------))
liquid : ((SP material)(ATTR (STATE *LIQUID)(SHAPE NIL)))

The descriptions (STATE)(MASS) and so on in the definition of 'material', lack values (V) showing that 'material' may have arbitrary values of these attributes. In the definition of 'liquid', there is an SP-link to 'material', which means that 'material' is a super-set concept of 'liquid', or that 'liquid' is a subset or a lower concept of 'material'. Objects belonging to a subset are considered to have the same properties as the objects of the super-set, in addition to the properties described explicitly in its definition.

By the above descriptions, we can see that the value of the attribute STATE of 'liquid' is *LIQUID, and that of SHAPE is the special value NIL. The *LIQUID is one of the primitive value markers. The primitive value markers are indicated by the preceding *. The value NIL indicates that 'liquid' can not have any value of SHAPE. By tracing up the SP-links, we can retrieve all the (A V)

pairs of an object. We assume the value of an attribute of a lower concept has precedence over that of the upper concept. For instance we can obtain the following full description of 'liquid'.

liquid : ((ATTR (STATE *LIQUID) (SHAPE NIL)(MASS)(COLOR) ---))

These upper-lower relationships among entity nouns are not expressed by a tree structure. Some nouns may share properties with more than one noun. 'Water' is such an example. 'Water' has the properties of both 'liquid' and 'compound'. Since we permit a noun to have several upper concepts, the relationships are represented by a lattice as shown in Fig. III-1.

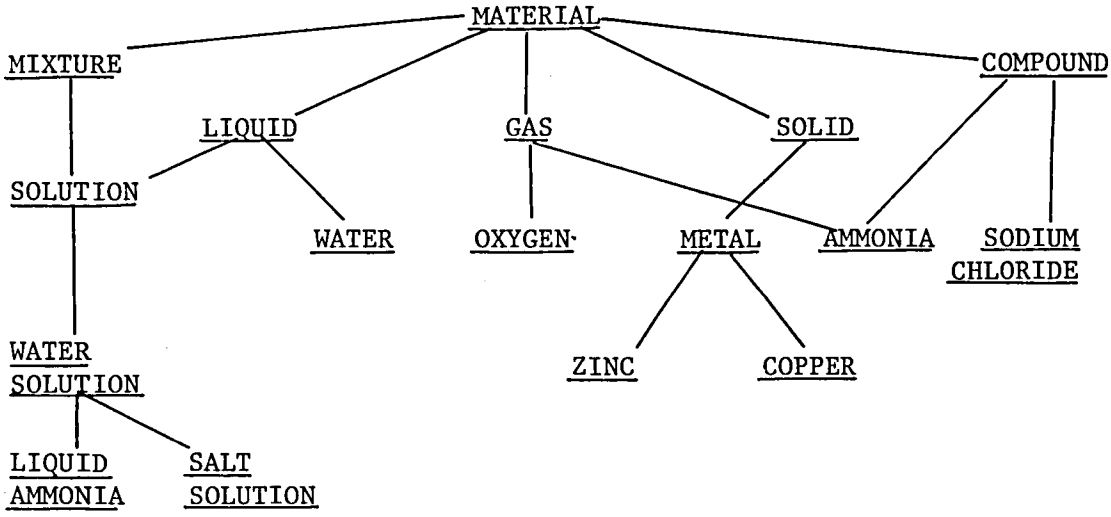


Fig. III-1 Upper-Lower Relationships among Nouns

Although most nouns are regarded as entity nouns, there are a few nouns which have relational functions. We call them *Relational Nouns*; 'Father' is a familiar example. In order to identify a person indicated by the word, we have to know whose father he is. In the chemical field we can easily find such nouns (e.g., 'weight', 'temperature', 'color', and 'mass'). These are called *Attribute Nouns*. Their meanings are described in a different way from that of ordinary

nouns. Fig. III-2 shows some examples. Here, A-ST designates the standard attributive relation which is expressed by the word. The description (NF N-A) shows the noun belongs to the group of attribute nouns.

```
( OOKISA      ( (NF  N-A)(A-ST   VOLUME MASS LENGTH AREA )
  size                                (SP  ZOKUSEI   RYO    )))
                                          attribute quantity
( IRO         ( (NF  N-A)(A-ST   COLOR)(SP  ZOKUSEI    SHITSU ) ))
  color                               attribute quality
```

* An attribute noun may express more than one standard attribute. OOKISA *(size)* expresses VOLUME, MASS, LENGTH or AREA. The attribute it expresses in context depends upon what entity noun is used with it.

** Attribute nouns are further classified into two groups, quantitative and qualitative. A qualitative attribute noun cannot be a case element of a verb which requires quantitative nouns. The verbs FUERU *(increase)* and HERU *(decrease)* are such examples of verbs.

Fig. III-2 Attribute Nouns

'Liquid' is another relational noun. The Japanese word which corresponds to 'liquid' is EKITAI. While 'liquid' in English can be either a noun or an adjective, EKITAI in Japanese is categorized syntactically as a noun. But semantically EKITAI has two different meanings, one corresponding to the noun usage of 'liquid', the other corresponding to the adjective usage of it. The noun EKITAI in the adjective usage is called a *Value Noun* with the attribute STATE. Another word AKAIRO *(red color)* is also a value noun of the attribute COLOR. Fig. III-3 shows the description of these nouns in the noun dictionary.

```
(EKITAI    ( (NF N-E)(SP  BUSSHITSU)(ATTR (STATE *LIQUID)(SHAPE NIL))
  liquid                  material
           ( (NF N-V)(V-DESCRIPTION (STATE * LIQUID)) )      )
```

```
(AKAIRO    ( (NF N-E)(SP  IRO ))
  red                   color
           ( (NF N-V)(V-DESCRIPTION  (COLOR *RED)))          )
```

Fig. III-3 Value Nouns

There are other kinds of relational nouns: *Action, Prepositional, Anaphoric* and *Function nouns*. An action noun is the nominalization of a verb. For example, KANSATSU *(observation)* is the nominalization of the verb KANSATSU-SURU *(observe)*. We describe this in the dictionary by giving a link to the original verb and by adding other information.

There are not postpositional particles in Japanese for every preposition in English. Some special nouns play the role of English prepositions. We call such nouns Prepositional Nouns. Because a prepositional noun usually has more than one meaning just as an English preposition has, we attach semantic conditions to help disambiguate them. Fig. III-4 shows examples of lexical descriptions of prepositional nouns.

```
( MAE                ( (NF N-P)(F-DESCRIPTION
  before
  in front of             ((CAT ACTION)      TIME BEFORE)

                          ((AND (CAT  N-E)(LOWER  DOUGU
                                                  instrument

                            BUSSHITSU))     PLACE  IN-FRONT-OF ))))
                             material

( NAKA               ( (NF  N-P)(F-DESCRIPTION
  in
                          (( OR  (LOWER  YOUKI     )(LOWER
                                         container
```

EKITAI)) PLACE IN))))
liquid

Fig. III-4 Prepositional Nouns

Corresponding to each meaning we give a triplet. The first element is the semantic condition. If the condition is satisfied, the corresponding second element is adopted as the meaning. If not, the next triplet is tried. The second element of a triplet represents the whole meaning of the phrase. For example, the whole meaning of the phrase TSUKUE [*desk*-entity noun] NO [*of*] UE [*on*-prepositional noun](on the desk) is PLACE. The third element of a triplet expresses the relationship by which the other noun in the phrase may specialize the whole meaning.

III-2-2 Verb Description

Verbs, adjectives, and prepositions in English have relational meanings with nouns. A verb represents a certain activity, while the agent associated with the activity is not inherent to the meaning of the verb (neither is the object which the activity affects, nor the other components). These components appear in a sentence with certain loose relations to a verb. In our system the meaning of a verb is described by setting up several relational slots which will be filled in by nouns. In this sense the meaning of a verb is not confined to itself, but is related to nouns.

We describe these relations by using the case concept introduced by C. J. Fillmore (1968). Case may be looked upon as a role which an object plays in an activity. Because several objects usually participate in an activity, there are several cases associated with an activity. An object is expressed by a noun phrase, and an

activity by a verb. A sentence instantiates an activity by supplying noun phrases to the cases associated with the activity. We call such instantiated activity an Event. The problem is to decide what case a noun phrase holds in relation to a verb in any particular event.

Though there are usually some syntactic clues in a sentence as to how it instantiates an activity, they are not enough to decide the case relationships between noun phrases and a verb. To establish these relationships we need both syntactic and semantic information. A verb has its own special usage patterns. That is, certain surface cases are necessary for the verb and certain objects are preferable as fillers for the case. We call these labeled patterns *SCS's* or *Case Frames* for Verbs, and express them as a list of case pairs such as (CASE NOUN). A verb usually has more than one case frame corresponding to different usages. A typical description of a verb is shown in Fig. III-5(The meanings of the symbols ACT, OBJ and so on will be given in pp 65 - 68).

```
( TOKASU          (CF
  melt
  dissolve        (( ACT  NINGEN      )( OBJ  KOTAI)(IN   EKITAI))
                          human being         solid       liquid

                  (( ACT  NINGEN      )( OBJ  KOTAI)(INST))
                          human being         solid

                  (( ACT  NINGEN      )( OBJ  KINZOKU)(INST SAN))
                          human being         metal          acid

                  (( ACT  SAN )( OBJ  KINZOKU  ))        ))
                          acid        metal
```

Fig. III-5 A Typical Description of a Verb

According to this description, we understand the surface verb TOKASU (*melt, dissolve*) has four different usages. In the first three

usages the verb takes the ACTOR case, and prefers to take the sub-concepts of the noun NINGEN (*human being*) as the case element. In such a way case frame descriptions are closely tied to noun descriptions, especially with the upper-lower concept relationships among nouns.

Notice that the verb TOKASU refers to two different activities. One refers to an activity which is usually expressed by the English verb *dissolve* and the other refers to an activity expressed by *melt*. We express these two activities by *DISSOLVE and *MELT respectively. *DISSOLVE and *MELT are the verbs in the DCS level. The first case frame is for the activity *DISSOLVE and the others are for the activity *MELT. We attached a DCS to each case frame. The meaning description of the verb 'TOKASU(*melt, dissolve*)' with DCS's is shown in Fig. III-6.

```
( TOKASU                (CF
  melt                  (( ACT  NINGEN      )( OBJ  KOTAI)(IN  EKITAI))
  dissolve                     human being        solid     liquid
                          ——►((*DISSOLVE  (ACT  IN)( OBJ  OBJ)))

                        (( ACT  NINGEN      )( OBJ  KOTAI)(INST)
                               human being         solid
                          ——►((*MELT  (ACT  /)( OBJ  OBJ)))

                        (( ACT  NINGEN      )( OBJ  KINZOKU)(INST SAN))
                               human being         metal          acid
                          ——►((*MELT  (ACT  INST)( OBJ  OBJ)))

                        (( ACT  SAN )( OBJ  KINZOKU)
                               acid        metal
                          ——►((*MELT  ((ACT  ACT)( OBJ  OBJ)))  ))
```

Fig. III-6 A Typical Description of a Verb with DCS's

There are two types of cases, *Intrinsic* and *Extrinsic* cases.

The intrinsic cases of a verb are essential ones for the activity, but extrinsic cases are not. For example, the cases of TIME and PLACE, which express when and where an event occurs, are extrinsic for ordinary verbs. Most activities can be modified by these extrinsic cases, but the kinds of nouns preferred for these case elements do not strongly depend on the kinds of activities. Therefore, we describe only the intrinsic cases in the verb dictionary. We set up following fourteen cases for the analysis of sentences in a textbook of elementary chemistry.

(1) ACT : ACTor is responsible for action.

(a) KARE-GA IOU -O SHIKENKAN-NI IRERU.
he-(ACT) *sulfur* -(OBJ) *test tube*-(IN, PLACE, ets.)*put in*
He puts sulfur in a test tube.

In the chemical field, a chemical object is often regarded as ACTor of an action, though it does not exercise intention in regard to action. For example, the underlined word in the following sentence is regarded as ACT.

(b) ENSAN -WA DOU -O TOKASU.
hydrochloric acid -*(all cases)* *copper* -*OBJ* *melt*
Hydrochloric acid melts copper.

(2) SUBJ : SUBJect is the primary topic of a sentence.

(a) KITAI-NO TAISEKI -GA FUERU.
gas *volume* -*(SUBJ)* *increase*
The volume of the gas increases.

(b) IOU -WA KIIROI.
sulfur -(SUBJ) *yellow*
Sulfur is yellow.

(3) OBJ : OBJect is the receiving end of an activity. It is affected by the activity.

(a) KARE-GA MIZU -O NESSURU.
he-(SUBJ) *water* -(OBJ) *heat*
He heats the water.

(b) ENSAN -GA AEN -O TOKASU.
hydrochloric acid -(ACT) *zinc* -(OBJ) *melt*

Hydrochloric acid melts zinc.

(4) IOBJ : This case is semantically the most neutral case. It is an object or concept which is affected by an activity, and which is not OBJect. This case is usually specialized by the other cases such as PLACE, TO, IN and so on, depending on the semantic interpretation of the verb itself.

(a) DOU -O ENSAN -NI TSUKERU.
copper -(OBJ) *hydrochloric acid* -(IOBJ) *dip*

(Someone) dips copper in hydrochloric acid.

(5) FROM : FORM describes a former position or state in time or space of the entailed SUBJect or OBJect of the verb.

(a) SHIKENKAN -KARA BIIKAA -E EKITAI-O UTSUSU.
test tube -(FROM) *beaker* -(PLACE) *liquid*-(OBJ) *pour*

(Someone) pours the liquid from the test tube into the beaker.

(6) RESULT : RESULT is to the future as FROM is to the past. It describes the resultant position or state as the entailed SUBject or OBJect of the verb.

(a) MIZU -GA SUIJOUKI -NI NARU.
water -(SUBJ) *steam* -(RESULT) *become*

The water becomes steam.

(7) INST : INSTrument is an object used as the tool or device by which an activity is carried out.

(a) GASU-BAANAA -DE MIZU -O NESSURU.
gas burner -(INST) *water* -(OBJ) *heat*

(Someone) heats water by a gas burner.

(8) TO : This is the destination in time or space of something in the action.

(a) SUIBUN -GA NAKUNARU TOKI MADE NESSHI TSUZUKERU.
water -(SUBJ) *be gone* *time*-(TO) *till heat continue*
(Someone) continues to heat (it) till the water is gone.

(9) FACT : FACT is used to indicate sentential complement.

(a) KORE -O SHITSURYOUHOZON-NO HOUSOKU -TO IU.
it -(OBJ) *the conservation of mass law* -(FACT) *call*
(We) call it the law of conservation of mass.

(10) PLACE : PLACE is used to indicate locations in space of the action.

(a) ARUCOORU-RANPU-NO YOKO -NI BIIKAA -O OKU.
alcohol lamp *side* -(PLACE) *beaker* -(OBJ) *put*
(Someone) puts a beaker on the side of an alcohol lamp.

(11) IN : IN indicates a more specific relation to PLACE.

(a) MIZU -O SHIKENKAN -NI IRERU.
water -(OBJ) *test tube* -(IN) *pour*
(Someone) pours water in a test tube.

(12) SOURCE : This shows constituent materials of compounds.

(a) ENSOSANNATORIUMU -WA ENSO, SANSO, NATORIUMU
sodium chlorate -(SUBJ) *chlorine* *oxygen* *sodium*
-KARA DEKITEIRU.
-(SOURCE) *consist*
Sodium chlorate consists of chlorine, oxygen and sodium.

(13) CAUSE : This shows a reason or cause of the activity.

(a) NESSHITA-TAME -NI HAGESHIKU KAGOUSURU.
heat -*reason* -(CAUSE) *violently* *react*
Because (someone) heats (them), (they) react violently.

(14) TIME : TIME indicates location in time of the action.

(a) NESSHITA -TOKI -NI SANSO -GA HASSEISURU.
heat *time* -(TIME) *oxygen* -(SUBJ) *be generated*
Oxygen is generated when (someone) heats (it).

In order to resolve some kinds of ambiguities, it is also necessary to utilize contextual information obtained from preceding sentences. When one knows a certain event has occurred, he can anticipate successive events that will occur and what changes the objects participating in the event will undergo. This kind of expectation plays an important role in understanding sentences. Various kinds of associations cluster conceptually around individual activities. One can perform contextual analysis of language by explicating these associations.

We append this kind of experiential knowledge to the DSC's of verbs. The following two items are described for each verb in the verb dictionary :

(1) CON : this refers to the consequent activities which are likely to follow the activity of the verb, but not necessarily.

(2) NTRANS : this refers to the resultant effects on objects in view of how the objects are influenced by the activity. In our system the influence on the objects is described by the following three expressions:

(a) (ADD case a-set-of-(A V)-pairs)

(b) (DELETE case a-set-of-attributes)

(c) (CREATE lexical-name-of-an-object a-set-of-(A V)-pairs)

(a) means that the object in the case indicated by the second element comes to have a set of properties indicated by the third element. (b) is for the deletion of a set of properties from the object. (c) shows that some objects will be created by the activity.

A typical example using a CON expression is shown in Fig.

III-7.

```
( IRERU  (CF
  put in
        ((( ACT  NINGEN        )(OBJ  BUSSHITSU)(IN    YOUKI))
                 human beings          material          container
         ⟶((*PUT   (OBJ  OBJ)(IN  IN))
            (CON (*EXIST (SUBJ  #OBJ))(PLACE  (IN  #IN))))))  ))
```

*The function # retrieves the designated case-element of the current DCS. *EXIST and *PUT are the verbs in the DCS level.

Fig. III-7 Example of Lexical Description

When we have completed the analysis of the sentence

IOU	-O	SHIKENKAN	-NI	IRERU.
sulfur	-(OBJ)	*test tube*	-(IN, RESULT, etc.)	*put in*

Someone puts sulfur in a test tube.

each case of the case frame of the verb IRERU (put in) is instantiated by an object referred to in the sentence, and therefore, the corresponding DCS is instantiated. Then we can instantiate the expression of CON, and the result is 'the sulfur is in the tube.' Fig. III-8 shows an example using an NTRANS expression.

```
(TOKASU       ( CF
 melt
dissolve
              ((( ACT   NINGEN       )(OBJ   KOTAI)(INST))
                        human beings        solid
                ⟶(( *MELT  (ACT /) (OBJ  OBJ))
                   (NTRANS  (ADD   OBJ (STATE  *LIQUID)))))

              ((( ACT   NINGEN       )(OBJ   KOTAI)(IN  EKITAI))
                        human beings        solid      liquid
```

```
——►((*DISSOLVE  (ACT  IN)(OBJ  OBJ))
    (NTRANS  (CREATE  YOUEKI
                      solution
                 ( SOLVENT  #ACT)
                 ( SOLUTE   #OBJ)))))    ))
```

Fig. III-8 Example of an NTRANS Expression

In this expression one can see the verb TOKASU has two different meanings. One corresponds to 'melt', and the other to 'dissolve in'. When we analyze the sentence,

DOU -O	TOKASU.
copper-(OBJ)	*dissolve, melt*

(Someone) melts copper.

We adopt the first case frame of TOKASU (melt) because it gives the highest matched value against the sentence (see section III-3-4). As the result of evaluating the NTRANS expression in the case frame, we conclude the copper is now in the liquid state. In the lexical description 'copper' is a lower concept of 'solid', so that copper in general behaves as a solid object. But the copper in the above sentence comes to have the attribute value pair (STATE *LIQUID) and will behave as 'liquid' in the succeeding sentences.

On the contrary, when we analyze the sentence

SHIO	-O	MIZU	-NI	TOKASU
salt	-(OBJ)	*water*	-(IN, PLACE, etc.)	*melt, dissolve*

(Someone) dissolves salt in water.

the second case frame of TOKASU (*dissolve in*) gives the highest matched value. After the sentence instantiates the case frame, a new object (i.e., a solution which consists of salt and water) will be created.

CON and NTRANS are thus important in the contextual analysis of sentences. The detailed analysis procedure using these expres-

sions will be described in Chapter IV.

III-3 Analysis of Noun Phrases

III-3-1 Properties of a Noun Phrase

In Japanese, two or more nouns are often concatenated by the postposition NO to form a noun phrase. Because there are many different semantic relationships among nouns concatenated by NO, we must decide what relationships may hold among the nouns. Typical examples are shown in Fig. III-9.

EKITAI -NO JOUTAI -NO SANSO -NO TAISEKI
liquid *state* *oxygen* *volume*

the volume of the oxygen in the state of liquid

HANNOU -NO ATO -NO NATORIUMU -NO TAISEKI -NO
reaction *after* *sodium* *volume*

HENKA
change

changes of the sodium's volume after the reaction

Fig. III-9 Examples of NOUN+NO Phrases

The phrase NOUN+NO can modify, in principle, any or all of the succeeding nouns in the extended NO construction so that many different patterns of modification relationships are syntactically permitted. We must decide which one is correct by considering semantic restrictions.

We have identified the following sixteen semantically acceptable NOUN NO NOUN combinations.

(1) (value noun)+(attribute noun)

(ex) KOTAI -NO JOUTAI
solid *state*

(2) (value noun)+(entity noun)

(ex) EKITAI NO IOU
liquid *sulfur*

(3) (entity noun)+(attribute noun)

(ex) EKITAI -NO IRO
liquid *color*

(4) (noun)+(prepositional noun)

(ex) HANNOU -NO MAE
reaction *before*

(5) (anaphoric noun)+(noun)

(ex) MOTO -NO BUSSHITSU
former *material*

(6) (attribute noun)+(entity noun)

(ex) (TAKAI) ONDO -NO EKITAI*
high *temperature* *liquid*

*In this usage, the attribute noun should be modified by another noun or adjective, which specifies the value of the attribute.

(7) (noun)+(action noun)

(ex) SANKADOU -NO KANGEN
oxidized copper *deoxidization*

IRO -NO HENKA
color *change*

(8) (time)+(noun)

(ex) (HANNOU -NO) MAE -NO DOU
reaction *before* *copper*

(9) (place)+(noun)

(ex) (SHIKENKAN -NO) NAKA -NO EKITAI
test tube *in* *liquid*

*The noun-noun combination, '*test tube*-NO *in*' expresses the 'place' in the test tube.

(10) (noun)+(conjunction noun)

(ex) SANKA -NO TAME*
oxidization *in order to*
by reason of

*In Japanese, some nouns are used to elucidate the case relationships between a noun phrase and verb. The noun TAME in this example expresses cases such as CAUSE or PURPOSE.

(11) (entity noun)+(entity noun)

(ex) NATORIUMU -NO KAGOUBUTSU*
sodium *compound*

*The first entity noun is a constituent element of the object expressed by the second noun.

(12) (entity noun)+(entity noun)

(ex) SANKADOU -NO SANSO*
oxidized copper *oxygen*

*The second noun is a constituent element of the object expressed by the first noun.

(13) (entity noun)+(entity noun)

(ex) SHIKENKAN -NO SOKO*
test tube *bottom*

*The second noun refers to part of the object expressed by the first noun.

(14) (entity nouns)+(entity noun)

(ex) KARIUMU, NATORIUMU -NADO -NO KINZOKU
potassium, sodium etc. *metal*

*The nouns *'potassium'* and *'sodium'* are lower concept nouns of the last noun *'metal'*.

(15) (name)+(noun)

(ex) SHITSURYOUHOZON -NO HOUSOKU
the conservation of mass *law*

(16) Others

(ex) $1cm^2$ ATARI -NO CHIKARA
per $1cm^2$ *pressure*

Corresponding to these relationships we prepared sixteen primitive functions. These functions are applied in turn to a noun phrase to decide what relationship holds between two nouns. The order in which these functions are applied is based on the frequency and the tightness of the relations. Each function checks only one semantic relation. In order to illustrate how these functions perform their tasks, the following example of 'noun + prepositional noun' phrase is given.

The noun MAE is a prepositional noun, and its semantic description is shown in Fig. III-4. We note that this word has two different meanings.

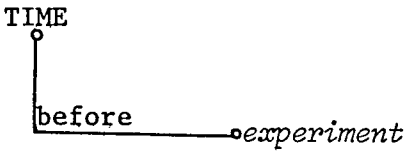
JIKKEN	-NO	MAE
experiment		time : *preceding*
		place : *in front of*

The function for the analysis of this kind of phrase checks at first whether the second noun MAE is a prepositional noun. If it is not, then this function fails and returns the value NIL. In this example, because the word MAE is a prepositional noun, the checking proceeds further. The description in Fig. III-4 shows that if the preceding noun is an action noun (i.e., if it is a nominalization of a verb) then MAE has the first meaning. Because the noun JIKKEN *(experiment)* satisfies this condition, the checking succeeds and the function returns the value T. The result of the analysis is shown in Fig. III-10(a). On the other hand, if the input is

TSUKUE	-NO	MAE
desk		*before, in front of*

then the word TSUKUE *(desk)* satisfies the condition of the second meaning, and the result is as shown in Fig. III-10(b).

(a)	JIKKEN	-NO	MAE
	experiment		*before* : time
			in front of : place



(b) TSUKUE -NO MAE
desk *before* : time
in front of : place

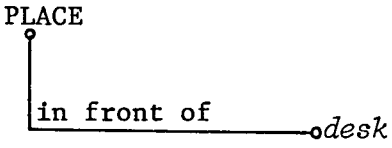


Fig. III-10 Results of Analysis of Noun and Prepositional Noun Phrase

In this way the sixteen checking functions not only test whether a certain semantic relationship holds among input words, but also disambiguates the meanings of input words.

III-3-2 Analysis Procedure for a Noun Phrase

We analyze a noun phrase by using the above sixteen checking functions subject to the limitation that related noun groups may not overlap. As stated before, 'noun + postposition NO' phrases and adjectives can modify only the succeeding nouns. We stack in the temporary stack noun phrases and adjectives for which the nouns to be modified have not been determined. The analysis of a noun phrase is carried out by scanning words one-by-one from left to right. If we scan an adjective or a determiner, we stack the word in the temporary stack and check whether it can modify the noun. This checking is done by the above functions if the stack word is a noun. We also have the checking functions relating nouns to adjectives or determiners. The dictionary content of an adjective is just the same as that of a value noun. The semantic checking function between an adjective and a noun will test whether the noun can have the attribute which is modifiable by the adjective. The checking of

the determiner differs somewhat and is explained in a later section.

The checking process will stop when there are no words in the temporary stack or a word is picked up that fails to modify the noun being scanned. The noun is then stacked in the temporary stack. If the temporary stack contains only one noun and there are no words to be scanned in the noun phrase, the analysis succeeds and returns the noun in the stack. The returned noun is called the Head Noun of the noun phrase. These processes are illustrated as follows.

SHIKENKAN	-NO	NAKA	-NO	AKAIRO	-NO	EKITAI
test tube		*in*		*red*		*liquid*

(i) Temporary Stack = empty

test tube -NO *in* -NO *red* -NO *liquid*

↑
scanned word

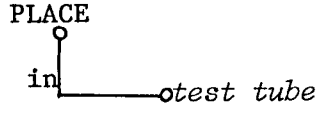
(ii) TS = | *test tube* |

test tube -NO *in* -NO *red* -NO *liquid*

↑
scanned word

*A check of the semantic relationship between *'test tube'* and *'in'* is performed.

**The phrase *'test tube* -NO in' is transformed into the form



(iii) TS = | *place* |

test tube -NO *in* -NO *red* -NO *liquid*

↑
scanned word

*A check of the semantic relationship between *'place'* and *'red'* is performed, but it failed to establish a new concept. Therefore, *'red'* is placed on the top of TS.

(iv) TS = | *red* | *place* |

test tube -NO *in* -NO *red* -NO *liquid*

scanned word

*The next scanned word is 'liquid'. Since it is a noun, a check of the relationship between the noun and the words in TS is performed. The check succeeds because the combinations (value noun)+(entity noun) and (PLACE)+(entity noun) are semantically permissible.

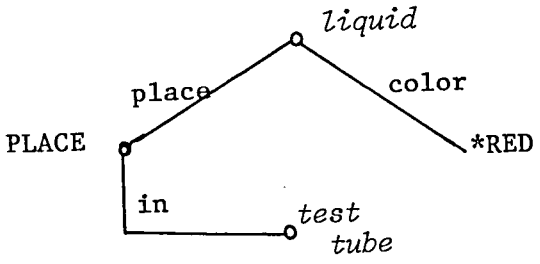
(v) TS = | *liquid* |

test tube -NO *in* -NO *red* -NO *liquid*

scanned word

*There are no words to be scanned, and the TS contains only one word. Hence, the analysis of this noun phrase succeeds.

**The result is as follows. (The head noun of this noun phrase is *'liquid'*.)



If there are no words to be scanned next and the temporary stack contains more than one word, then the analysis fails and backtracks to the decision points of the program. A decision point in the analysis of a noun phrase is any point at which two words have been related semantically. The relationship between two words established during the analysis is the one determined by the function which succeeds first. Because the order of checking functions is somewhat arbitrary, in some cases a relationship which has not been checked may be preferable to the established relationship. This is illustrated in the examples below.

EKITAI	-NO	JOUTAI	-NO	HENKA
liquid		*state*		*change*

the change of state of the liquid

EKITAI	-NO	JOUTAI	-NO	SANSO
liquid		*state*		*oxygen*

oxygen in the liquid state

In the first example the word JOUTAI (*state*) designates an attribute of EKITAI (*liquid*) and EKITAI corresponds to a visible, real object. JOUTAI (*state*) in the second example disignates an attribute of SANSO (*oxygen*), and the word EKITAI does not correspond to a real object but is used to specify the attribute 'state' of the oxygen. These examples show that the word EKITAI (*liquid*) has two different usages. According to these usages, there are two different semantic constructions of the phrase EKITAI-NO JOUTAI as shown in Fig. III-11.

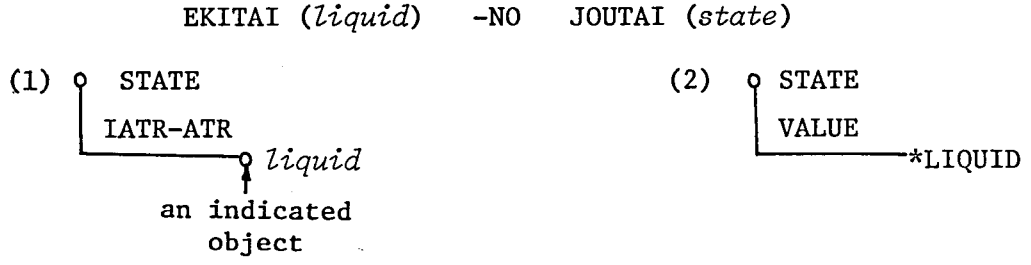
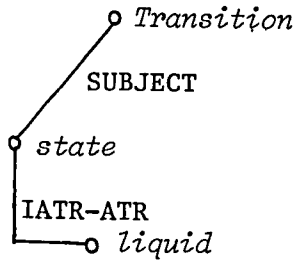


Fig. III-11 Two Different Deep Structures for the Phrase EKITAI NO JOUTAI

Because we analyze a noun phrase from left to right, we cannot determine which usage is correct until we recognize the rightmost word HENKA (*change*, *transition*) or SANSO (*oxygen*). However, a semantic checking function disambiguates the multiple meanings of the word EKITAI. If the disambiguation is recognized to be incorrect in subsequent processing, we must be able to backtrack to the decision point at which this temporary disambiguation was made. We implemented such a process by using PLATON's backtracking facilities. This process is illustrated as follows.

Input ; EKITAI -NO JOUTAI -NO HENKA
liquid *state* *transition*

result :



Input ; EKITAI -NO JOUTAI -NO SANSO
liquid *state* *oxygen*

Steps of Analysis:

(i) TS = empty

liquid -NO *state* -NO *oxygen*
↑
scanned word

(ii) TS = *liquid*

liquid -NO *state* -NO *oxygen*
↑
scanned word

(iii) TS = *state*

liquid -NO *state* -NO *oxygen*
↑
scanned word

*At this point, the first meaning of '*liquid*' has been adopted because the checking function for (entity noun)+(attribute noun) is applied before the function for (value noun)+(attribute noun). That is, the word '*liquid*' indicates a physical object.

**The semantic check between '*state*' and '*oxygen*' fails, because the attribute noun '*state*' has been linked to the liquid by the relation IATR-ATR and an attribute noun cannot be linked with two different entity nouns.

***So the program will go back to step (ii).

(iv) TS = *liquid*

liquid -NO *state* -NO *oxygen*
↑
scanned word

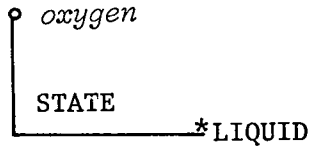
*The semantic check between '*liquid*' and '*state*' proceeds further. The semantic checking function for (value noun)+(attribute noun) succeeds. This function adopts the second meaning of '*liquid*'.

(v) TS = *state*

liquid -NO *state* -NO *oxygen*

scanned word

*At this time, because the noun '*state*' is only linked to the value *LIQUID, the check between '*state*' and '*oxygen*' succeeds. The result is as follows. Notice that the noun '*liquid*' does not express a real object but the value of the attribute '*state*'.



III-3-3 Analysis of Conjunctive Noun Phrases

The words in Japanese which correspond to 'and' and 'or' are categorized as special postposition ; some of them are shown in Table III-1. We call them Conjunctive Postpositions.

postposition	corresponding English
TO	and (closed listing)
YA	and (open listing)
MO	and (also)
KA	or
. . .	. . .

TABLE III-1 Conjunctive Postpositions in Japanese

In Japaneses as well as in English, it is difficult to determine the scope of a conjunction. There are some phrases which have the same syntactic structure, but semantically different constructions. Some examples are shown in Fig. III-12. On the other hand, some phrases have different surface structures but convey the same meaning as is illustrated in Fig.III-13. As there are few syntactic clues in these examples, we must analyze them by using semantic information.

At the first stage of the analysis of a noun phrase, we try to find conjunctive postpositions. If we cannot find them, the normal analysis sequence described above is applied to the noun phrase. If there is a conjunctive postposition, the following steps are performed:

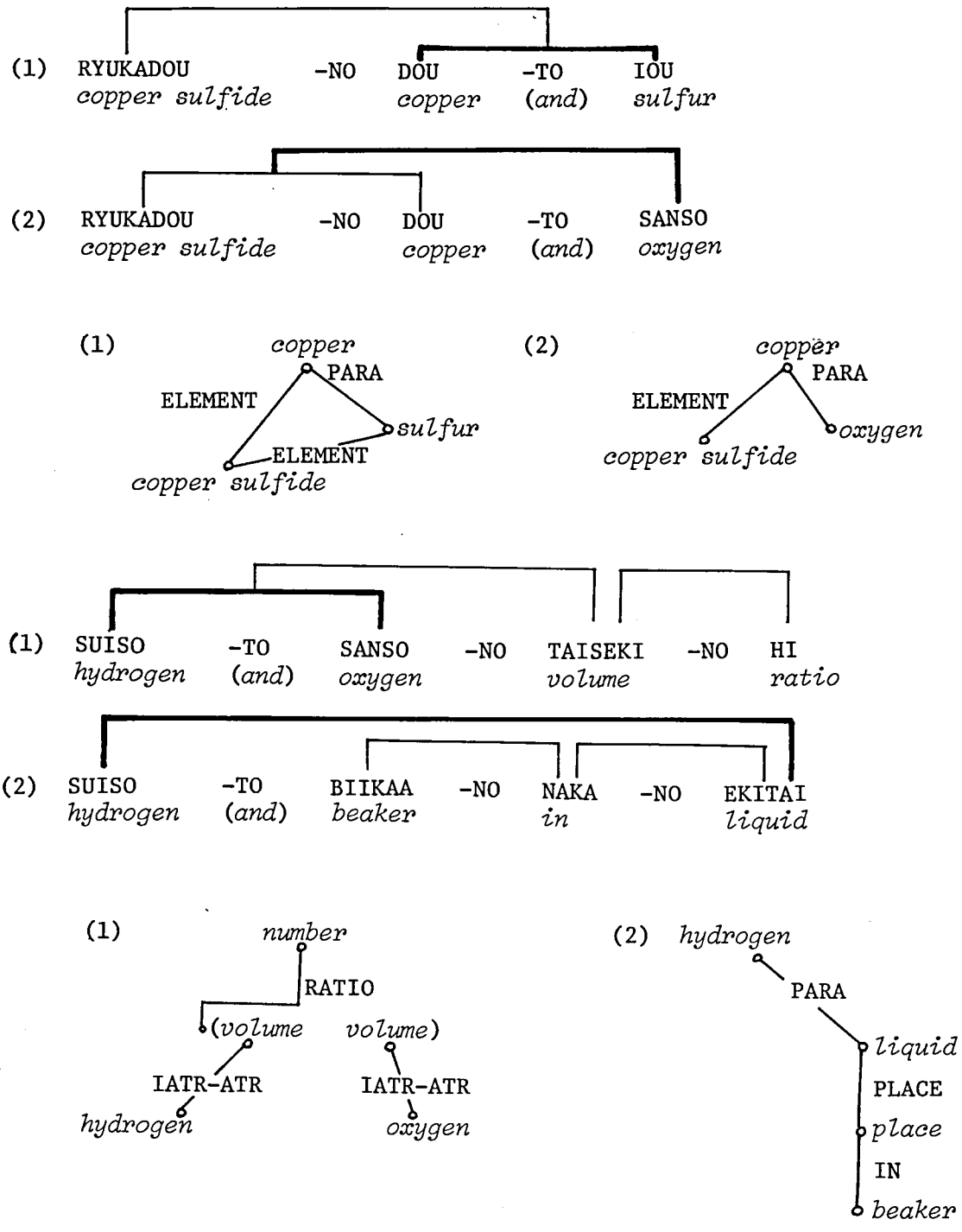
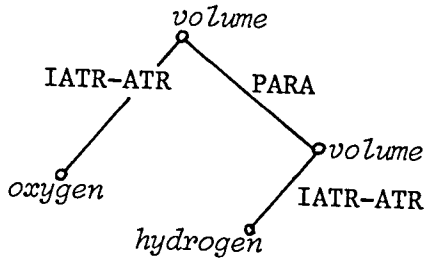
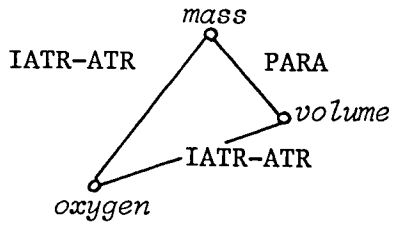


Fig. III-12 Examples of Conjunctive Phrases
(The meanings of symbols will be given in p 95.)

(1)	SANSO *oxygen*	-NO	TAISEKI *volume*	-TO (*and*)	SUISO *hydrogen*	-NO	TAISEKI *volume*	-(TO)*
(2)	SANSO *oxygen*	-TO (*and*)	SUISO *hydrogen*	-NO	TAISEKI *volume*			



(1)	SANSO *oxygen*	-NO	SHITSURYOU *mass*	-TO (*and*)	SANSO *oxygen*	-NO	TAISEKI *volume*	-(TO)
(2)	SANSO *oxygen*	-NO	SHITSURYOU *mass*	-TO (*and*)	TAISEKI *volume*	-(TO)		



*(TO) is an optional element in these sentences.

Fig. III-13 Examples of Differing Surface Structures Conveying the Same Meanings

Step 1. The conjunctiye postposition TO is often followed by another postposition TO in the succeeding part (Fig. III-13). Hence if we find TO in the phrase, we do the following; if not, go to Step 2. We search for the second postposition in the succeeding part. If it is found, then the noun phrase before the first postposition and the noun phrase interposed between the first and the second postpositions are paralleled. We employ the normal noun phrase analysis to the interposed noun phrase, then go to Step 4. If we cannot find the second postposition, we then go to Step 2.

Step 2. If a conjunctive postposition is not TO, or there is no second TO, we execute the following substeps. (Noun-1 designates the noun before the first postposition.)

a. Search for a noun identical to Noun-1 in the succeeding part. If found, let it be Noun-2, and go to Step 3.

b. If Noun-1 is not an entity noun, then search for a noun which belongs to the same category as Noun-1. If found, let it be Noun-2, and go to Step 3.

c. Search for a noun which has an upper concept in common with Noun-1. If found, let it be Noun-2, and go to Step 3.

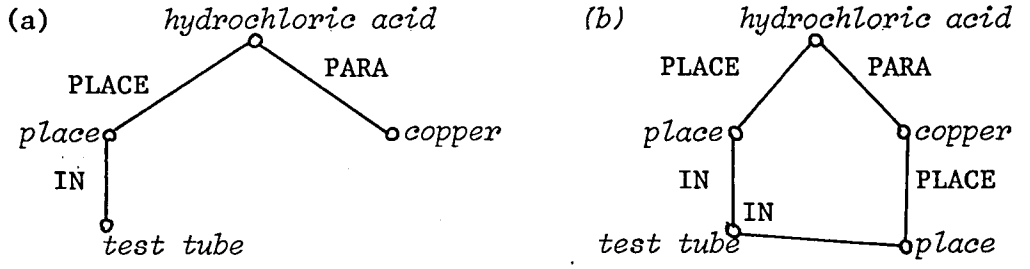
Step 3. The phrase between the postposition and Noun-2 are analyzed by the normal noun phrase analysis. This is now the second of the two parallel phrases under consideration.

Step 4. The phrase before the postposition is analyzed by the normal noun phrase analysis.

Step 5. It is necessary to determine what portion of the phrase before the postposition relates exclusively to Noun-1. To determine the left end of the Noun-1 phrase (e.g., in Fig. III-14 below), we pick words one-by-one from left to right, and check whether each word can modify Noun-2. The first word found which cannot modify

Noun-2 is considered the left end of the first phrase (Noun-1 phrase).

SHIKENKAN -NO NAKA -NO ENSAN TO DOU -(TO)
test tube *in* *hydrochloric acid* *copper*



the hydrochloric acid in the test tube and the copper

the hydrochloric acid and copper in the test tube

Fig. III-14 Two Different Constructions According to the Two Different Determinations of the Left End of the Conjoined Phrase

Step 6. Words to the right of Noun-2 are checked to determine their relation to the conjunctive phrase and its conjuncts. Checking proceeds from left to right.

The analysis of the following phrase is illustrated in Fig. III-15(See the next two pages).

RYUKADOU -NO DOU -TO IOU -NO SHITSURYO -NO HI
copper sulfide *copper* (*and*) *sulfur* *mass* *ratio*

the ratio between the mass of the copper and the sulfur which constitute copper sulfide

III-4 Analysis of a Simple Sentence

Japanese is a typical SOV language in which ACTOR, OBJECT and other case elements usually appear before the verb. The construction of a typical Japanese sentence is shown in Fig. III-16.

PHRASE:

RYUKADOU -NO DOU -TO
copper sulfide *copper* (conjunctive pp---*and*)

IOU -NO SHITSURYOU -NO HI
sulfer *mass* *ratio*

meaning: *the ratio of the masses of copper and sulfur of copper sulfide*

sequence of analysis:

(1) Step 1. Find the conjunctive postposition TO.

RYUKADOU -NO DOU TO IOU -NO SHITSURYOU -NO HI
copper sulfide *copper* *sulfur* *mass* *ratio*

former part *succeeding part*

(2) Step 2c. Find from the succeeding part the noun which belongs to the same category as *'copper'*. In the above phrase, the noun *'sulfur'* is found.

RYUKADOU -NO (DOU -TO IOU -NO) SHITSURYOU -NO HI
copper sulfide *copper* *mass* *ratio*

temporarily determined scope of the conjunctive phrase

(3) Step 3. not applicable

(4) Step 4. Analyze the phrase before the postposition TO.

(DOU - TO IOU -NO) SHITSURYOU -NO HI
copper *sulfur* *mass* *ratio*

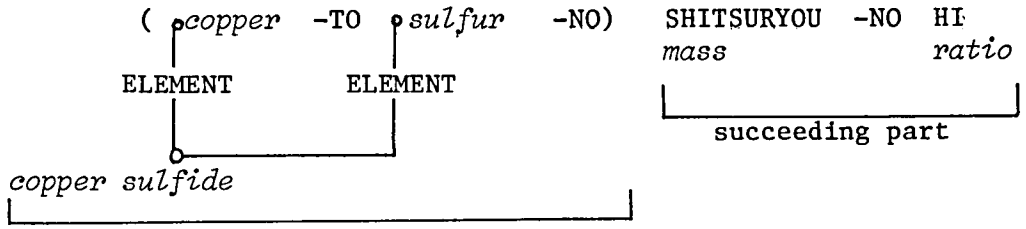
ELEMENT

copper sulfide

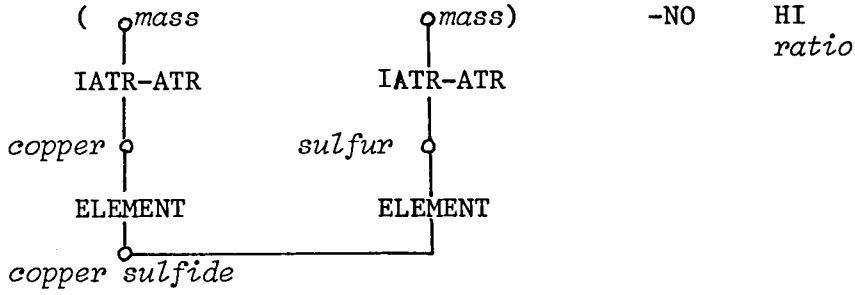
(5) Step 5. The second noun of the conjunctive phrase, *'sulfur'* is checked against the leftmost noun of the phrase before the postposition *'copper sulfide'*. This noun is related to the first noun of the conjunctive phrase, *'copper'*. *'copper sulfide'* is also

Fig. III-15 Example of the Analysis of Conjunctive Phrase

seen to be related to *'sulfur'*. This places the left boundary of the Noun-1 phrase immediately to the left of *'copper'*.



(6) Step 6. The two nouns, *'copper'* and *'sulfur'*, in the conjunctive phrase are checked against nouns in the portion following Noun-2. Because the noun *'mass'* can be related to only individual physical objects, the noun *'mass'* is duplicated for *'copper'* and *'sulfur'*.



The noun *'ratio'* is related to the conjunctive phrase as a whole. Hence, we obtain the following result for the entire conjoined phrase.

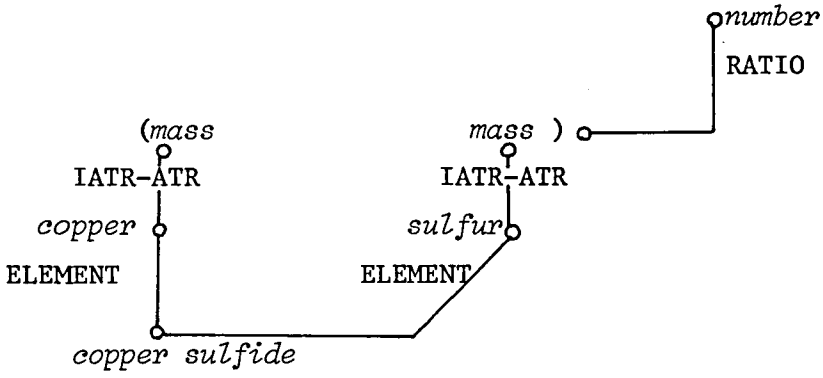
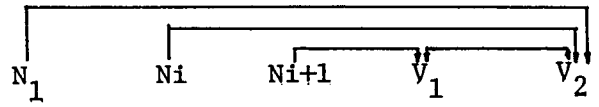


Fig. III-15 continued



(ex 1) KARE -WA GASU-BAANAA -DE SHIKENKAN -NI IRETA
He (all cases) *gas burner* (INST) *test tube* (PLACE) *put-in*

relative clause

EKITAI -O NESSURU.
liquid (OBJ) *heat*

meaning: *He heats the liquid which is put in the test tube.*

(ex 2) ENSOSAN-NATORIUMU -O GASU-BAANAA -DE (ENSOSAN-NATORIUMU-O)*
sodium chlorate (OBJ) *gas burner* (INST) *sodium chlorate* (OBJ)

SHIKENKAN -NI IRETE NESSURU.
test tube (IN, PLACE, etc.) *put in* *heat*

meaning: *(someone) puts sodium chlorate in a test tube and heats it.*

(*)Usually this phrase is omitted.

Fig. III-16 Typical Japanese Sentences

A verb may govern several noun phrases -- case elements -- preceding it. A relative clause modifying a noun may appear in the form '--- verb + noun ---'. The right end of the scope of the clause is easily identified by finding the verb, but the left end is harder to identify. In Fig. III-16 the noun phrase NP_{i+1} is a case element of verb V_1. The noun phrase NP_i is governed by verb V_2. Because the rule of projections holds (i.e., clauses do not overlap) in Japanese as in other languages, all noun phrases between NP_{i+1} and V_1 are

governed by V_1, and the noun phrases before NP_i are governed by V_2. However, boundaries as between NP_i and NP_{i+1} cannot be determined uniquely by syntactic clues alone. To determine them we must use semantic relationships such as case relationships between noun phrases and verbs.

In English case in surface structure is generally evident in the order of phrases. In Japanese, the ordering of noun phrases is relatively free. A postposition attached to a noun phrase usually shows the case which the noun phrase plays in the sentence. The postpositions usually used in Japanese and the surface cases in the case frames(SCS) corresponding to them are tabulated in Table III-2.

postposition	case
-GA	ACT, SUBJ
-NO	NMOD (ACT, SUBJ)
-O	OBJ
-NI	RESULT, IN, IOBJ, TO, PLACE, CAUSE, TIME
-HE	TO
-TO	FACT, RESULT, TAISHO
-KARA	FROM, SOURCE, CAUSE, METHOD, PLACE, TIME
-YORI	FROM, SOURCE
-DE	INST, SOURCE, CAUSE, METHOD, PLACE, TIME
-MADE	TO
-WA	all cases
-DAKE	
-MO	all cases
-SHIKA	
. . . .	

TABLE III-2 Postpositions in Japanese and Case Relationships

From this table one can see that a postposition in surface structure does not necessarily correspond to a unique case. In the course of analysis we must assign appropriate case labels by considering the case frames of the main verb along with meanings of the head nouns of the noun phrases.

A postposition also plays the role of a delimiter which shows the right boundary of a noun phrase. The outline of the analysis of a simple sentence is as follows:

(1) At first the program looks for a verb in the input sentence. Because there may be embedded sentences which modify nouns in the main sentence, there is usually more than one verb in the input sentence. The program picks up the leftmost verb of the sentence.

(2) The string before the verb is segmented by locating postpositions.

(3) Since each segment is assumed to constitute a noun phrase, each is passed to the program which analyzes noun phrases.

(4) When all the segments are analyzed and the head nouns are determined, the program checks each noun phrase against the verb asking whether a case relationship will be satisfied between the noun phrase and the verb. The checking is carried out right to left starting with the phrase nearest to the verb.

(5) When there are no more noun phrases to be checked, or when a noun phrase which cannot be a case element of the verb is found, the checking is terminated. If there remains an intrinsic case slot of the verb which has not been filled, we search for an appropriate noun to fill the slot from the context. This searching process will be explained in Chapter IV. We determine whether a noun phrase can be a case element of a verb by the following syntactic and semantic clues:

(1) The type of postposition which follows the noun phrase. This marks case in the surface structure.
(2) The case frames of the verb.
(3) The meaning of the head noun of the noun phrase.

The postposition delimits a set of possible cases by which the noun phrase may be related to a verb. We must choose an appropriate one from this set by using the second and third types of information. The case slot fillers in a case frame of a verb are relatively upper

concept nouns. A sentence is considered to be an instantiation of a case frame, and the nouns employed will generally be lower concept nouns of the nouns in the case frames.

Suppose we analyze the sentence:

SHOKUEN -O MIZU -NI TOKASU.
salt (OBJ) *water* (IN, RESULT, TIME, etc.) *melt, dissolve*

(Someone) dissolves salt in water.

We can check whether the sentence matches the case frame of TOKASU:

TOKASU : ((ACT human)(OBJ material)(IN liquid)
melt
dissolve

The checking is performed by considering whether '*salt*' is a lower concept noun of '*material*', and whether '*water*' is a lower concept noun of '*liquid*'.

Because a case frame contains only intrinsic cases of a verb, we check extrinsic ones when a noun phrase is found not to be an intrinsic case element of the verb. That is, we check whether the postposition can mark the TIME or PLACE, and whether the noun phrase is an instance of the noun 'place' or 'time'.

The above process may appear straightforward. But sentences can have several possible interpretations for the following reasons.

(1) A verb may have more than one usage (i.e., a verb may have several possible case frames).

(2) A postposition can indicate more than one case. Some postpositions can occur with almost any case; WA is an example.

(3) A noun modified by an embedded sentence is usually a case slot filler of the embedded sentence. But we may have no syntactic clues as to what case to assign to the noun.

In the event of multiple interpretations the program derives labeled interpretations showing all possible case relationships between specific nouns and verbs. We choose the interpretation showing the preferable matching of nouns and case by using an evaluation function

below which has been established expirically.

$$f(CFN, C1, C2, C3) = \frac{6 \times C1 + 2 \times C3}{CFN} + \frac{C2}{2}$$

CFN : number of intrinsic cases in a case frame

C1 : number of intrinsic case elements which are filled by noun phrases in the sentence.

C2 : number of extrinsic case elements which are filled by the noun phrases in the sentence.

C3 : number of intrinsic case elements which are filled by the noun phrases in the preceding sentences.

The value of this function indicates the degree of matching between a sentence and the case frame of the verb in question. The trial frame which gives the highest matched value is selected. We then proceed to the analysis of the remaining strings. If the selection is found to be wrong during the succeeding analysis, control comes back to the point at which the decision was made, discards it, and chooses the pattern which gives the next highest matching value.

CHAPTER IV

CONTEXTUAL ANALYSIS OF JAPANESE SENTENCES

IV-1 Introduction

Our view of the process of sentence understanding is roughly as follows. One reads sentences from left to right and understands them in succession. When he cannot understand a sentence satisfactorily he refers back to the preceding sentences to obtain a key to understanding. If he cannot find what is needed, he leaves the question pending and proceeds to the next sentence. If a phrase or a sentence is found which seems to solve the question, then he checks whether it can really resolve the question. If so the sentence is properly organized into the previous context and the question is dismissed. In any case the pending question is likely to be dismissed as time passes.

We feel this process of sentence understanding is not especially complex. It can be realized through an artificial intelligence approach. While we recognize that some kinds of problems may be solved only by using complicated logical operations, we think most problems in understanding sentences can be solved by relatively simple operations. Logical operations may only be effectively applied on a complete data base in which all the necessary axioms (corresponding to human knowledge) are declared and no contradictory axioms exist. In the course of reading sentences, one has only partial knowledge about the context, and therefore, his knowledge is not complete. However, a person understand the meaning of sentences before he reads through the entire set. This means that one is sometimes content with incomplete deductions for understanding sentences. For this

reason, rather than logical operations we employ heuristically admissible operations which use an intermediate term memory structure and semantic relationships described in the dictionary.

We conceive of three types of memory. Long term memory incorporates knowledge of the world, not considered here. Short term memory is for immediate recall of unanalyzed strings under consideration. Intermediate term memory is limited but contains a structured representation of recently analyzed strings and strings under analysis. We summarize our approach as follows:

(1) Context is entered into the intermediate term memory.

(2) Two kinds of intermediate term memory are prepared. One is for representing the current contextual content, and the other is to sustain pending questions. The former is further divided into the noun stack (NS) and the hypothetical noun stack (HNS). The latter is called the Trapping List (TL).

(3) Contextual analysis is performed after the processing of each syntactic unit such as a noun phrase or a sentence which conveys a unitary idea.

(4) NS is organized such that theme words of sentences can be easily retrieved. Here 'theme words' mean the key subjects mentioned in the sentences.

(5) Sometimes we have to refer to the succeeding sentences in order to understand a sentence. In such cases we do not immediately refer to the succeeding sentences, but instead hold a pending question in TL to be resolved in the course of analyzing the succeeding sentences.

IV-2 Memory Structure for Contextual Information

The analysis of a sentence is primarily grounded in the

semantic description -- case frame -- of a main verb. Contextual analysis is mainly grounded in accumulated information about nouns. The objects or concepts that are the themes of the sentences, and what has been predicated of them can usually be characterized in terms of the nouns appearing in the sentences, and these offer important clues for contextual analysis.

We assign a different LISP atom (produced by the LISP function 'gensym') to each noun which appears. Information about each is entered on the respective property list. The flags tabulated in Table IV-1 are used.

relation	content
LEX	link to the dictionary lexical descriptions
SATR	(A V) pairs which specify this object
CASE	link to the case-frame in which the object appears
PRE	link to any noun atom which appears in the previous sentence, and which represents the same object as this atom
POST	the inverse relation of PRE
SMOD	link to any relative clause which modifies this object
PARA	link to any noun atoms which appear in a conjunctive phrase together with this object

Table IV-1 Information Attached to a Noun Atom$_1$

We can retrieve all the descriptions given for an object to which a noun has been assigned. We stack these LISP atoms called Noun Atoms on NS and HNS.

IV-2-1 Noun Stack (NS)

When we start to analyze a sentence, we stack a list of noun

atoms which are assigned to the nouns in the sentence. These noun atoms are re-ordered according to their degrees of importance. NS has the construction shown in Fig. IV-1.

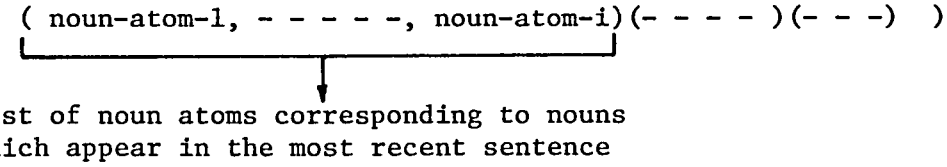


Fig. IV-1 Construction of NS

To decide how important a noun is, we use the following heuristics.

(1) In Japanese a theme word is often omitted or expressed by a pronoun in succeeding sentences after it appears once. In other words, the word which is omitted or expressed by a pronoun is an important word for the understanding of a sentence.

(2) A theme word may appear as SUBJ in the surface case structure. To emphasize a word which is OBJ-case in deep case structure, or to de-emphasize a word in the ACT-case which is not worth mentioning, the passive voice may be used. This places a stressed word in the subject position of the sentence which would otherwise appear as object or indirect object.

(3) The importance of a head noun in a noun phrase is greater than that of other nouns.

A simple example of ranking by importance is shown in Fig. IV-2; zinc appears in all the sentences and is the theme word.

Input sentence:

N1			N2	N3
RUTSUBO	-NI		100gr-NO	SHITSURYOU-NO
melting pot	(PLACE, TIME, IOBJ, etc.)			*mass*

N4			N5	
AEN	-O	IRETE ,	GASU-BAANAA	-DE
zinc	(OBJ, IOBJ)	*put in*	*gas burner*	(PLACE, INST, etc.)

NESSHI, TOKASHITA.
heat *melt* (PAST TENSE)

Meaning of the input sentence:

S1: *(Someone) put 100g of zinc in a melting pot.*

S2: *(Someone) heated it by a gas burner.*

S3: *(Someone) melted it.*

Changes of NS

Beginning of the analysis of S1: ((N4 N3 N2 N1))

End of the analysis of S1: ((N4 N1 N3 N2))

Beginning of the analysis of S2: ((N5)(N4 N1 N3 N2))

End of the analysis of S2: ((N4 N5)(N4 N1 N3 N2))

Beginning of the analysis of S3: (NIL (N4 N5)(N4 N1 N3 N2))

End of the analysis of S3: ((N4)(N4 N5)(N4 N1 N3 N2))

Fig. IV-2 Changes of NS

We first show examples which cannot be properly analyzed without HNS.

(a) SUISO -TO SANSO -O 2:1 -NO WARIAI -DE KONGO-SHI,
hydrogen *oxygen* (OBJ) *two to one* *ratio* *intermix*

KONO KONGOUKITAI -NI - - - - -
this *gas mixture* (PLACE) - - - - -

(Someone) intermixes hydrogen and oxygen in the ratio of two to one. - - - - - in this gas mixture - - - - -

(b) SHOKUEN 5gr -O MIZU 100cc -NI TOKASU.
salt *five grams* (OBJ) *water* (IN) *dissolve*

KONO SUIYOUEKI -WA - - - - -
the *solution*

(Someone) dissolves 5 grams of salt in 100cc of water. The solution is - - - - -

In these two examples, though the demonstrative KONO (the, this) is used, the object referred to does not appear explicitly in the preceding sentence. The object referred to is produced as the result of the event which is expressed by the preceding sentence. As mentioned before, we append to case frames in the verb dictionary descriptions of any objects which may be created if the verb is used.

TOKASU (*dissolve*) has the case frame:

((ACT human)(OBJ material)(IN liquid))

corresponding DCS is:

(*DISSOLVE (ACT IN)(OBJ OBJ))

and this DCS has the additional description:

```
(NTRANS  (CREATE 'solution  ('solvent (# ACT))
                            ('solute  (# OBJ))  )).
```

The symbol # in this description is a LISP function which fills the specific case elements indicated in the argument from the current realization of the case frame. The sentence

SHOKUEN	5gr	-O	MIZU	100cc	-NI	TOKASU.
salt		(OBJ)	*water*		(IN)	*dissolve*

associated with the above case frame results in the following interpretation: a new object, a solution whose solvent is water and whose solute is salt results. We represent this newly produced object in HNS instead of NS for the following two reasons.

1. As the description is based on uncertain knowledge, it is likely, but not necessarily so that the object is produced in the real world. If we find some descriptions of this derived object in the succeeding sentences, we will decide it really exists and transfer the representation from HNS to NS.
2. Because the newly produced object is referred to in the succeeding sentences sometimes by different words or by syntactically different forms, it is convenient to stack them individually in HNS.

IV-3 Estimation of the Omitted Words

In the analysis of a Japanese sentence it is important to supply omitted words drawing from preceding or succeeding sentences. To do this we must be able to:

1. recognize that a word is omitted and
2. search for an appropriate word to fill the gap.

Our contention is that an individual syntactic unit such as a noun phrase or a simple sentence conveys a definite idea; a noun phrase may designate a certain definite object, a concept, or whatever, and a simple sentence may describe a definite event. In order that a

simple sentence describes a definite event, each intrinsic case element of the case frame must be specified by particular objects. We can detect an omitted word by searching for unspecified case elements in a case frame. Moreover, we can guess from the case frame what kind of nouns should be supplied to fill any gaps.

In this manner we can detect and supply omitted words by using the semantic descriptions in the dictionary.

IV-3-1 Omitted Word in a Simple Sentence

When we have finished the analysis of a simple sentence, we check whether there remain some intrinsic cases to be specified. If there remain some, we search for appropriate fillers in the preceding sentences. The searching process is carried out in the following way.

(i) We search through HNS first, because an object newly created by the preceding event is often the theme object of the present event.

(ii) In Japanese, identical case elements in succeeding sentences are apt to be omitted. So the previous sentence is searched for elements having the same case relation as the one under consideration through NS.

(iii) If the above processes fail, then we check the words in NS or all the words that have appeared in the three previous sentences one-by-one until we find a semantically admissible word.

(iv) If we cannot find a suitable word, we set up a problem in the trapping list TL (mentioned in the next section).

Some results of the processing are shown in the followings.

(a) Example 1

Input sentence:

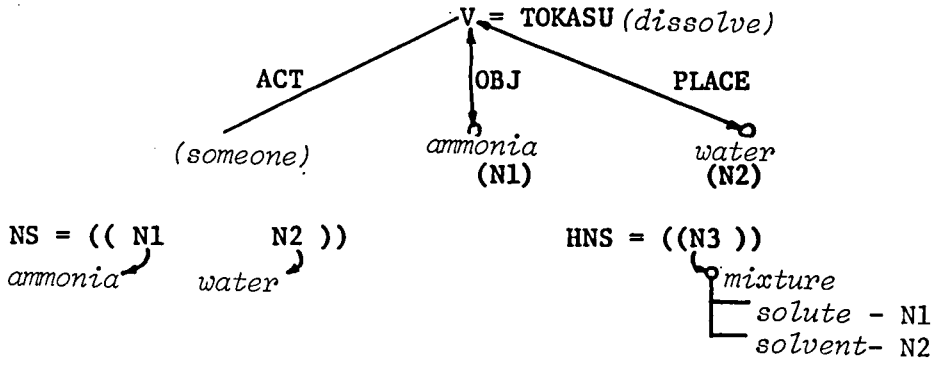
AMMONIA	-O	MIZU	-NI
ammonia	(OBJ)	*water*	(PLACE, TIME, IOBJ, etc.)

TOKASHI,		RITOMASUSHI	-O	TSUKERU.
dissolve, melt		*litmus paper*	(OBJ, IOBJ)	*soak, put, etc.*

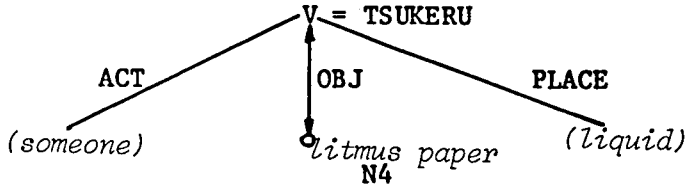
meaning: *(Someone) dissolves ammonia in water and puts litmus paper (in it).*

Analysis Process:

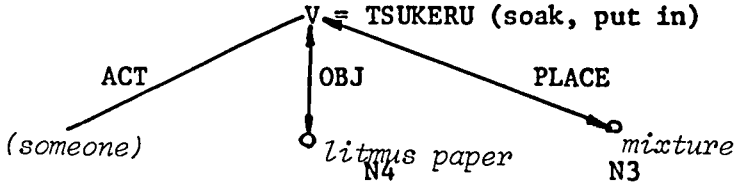
*result of the analysis of the first sentence



*intermediate result of the analysis of the second sentence



*final result obtained after searching



(b) Example 2

Input sentence:

NAFUTHARIN -O SHIKENKAN -NI
naphthaline (OBJ, IOBJ) *test tube* (PLACE, IOBJ, IN, etc.)

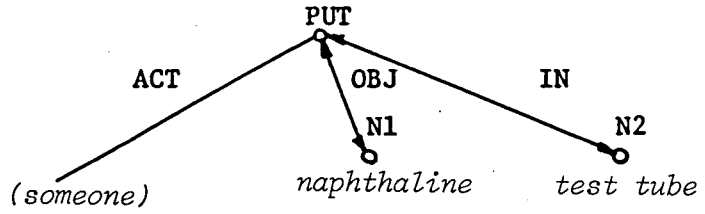
IRE, GASU-BAANAA -DE NESSHITE, TOKASHI,
put in *gas burner* (INST, METHOD) *heat* *melt*

KANSATSUSURU.
observe

meaning: *(Someone) puts naphthaline in the test tube.*
(Someone) heats (it) by a gas burner.
(Someone) melts (the naphthaline).
(Someone) observes (the naphthaline).

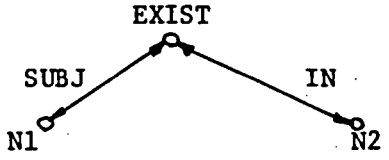
Analysis Process:

*result of the analysis of the first sentence

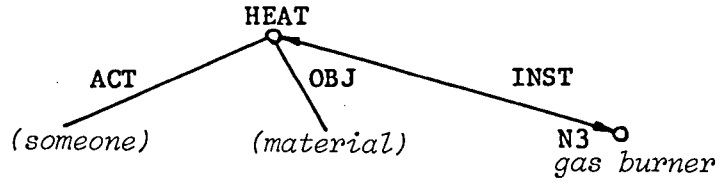


NS = ((N1 N2)) HNS = NIL

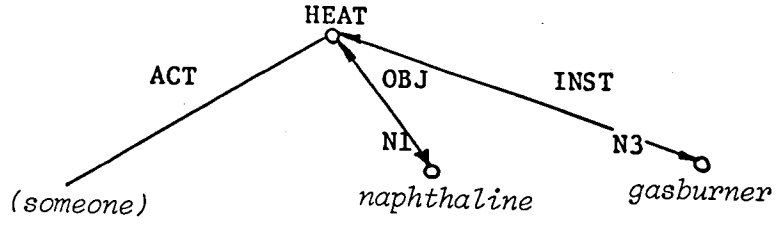
temporary assertion:



*intermediate result of the analysis of the second sentence

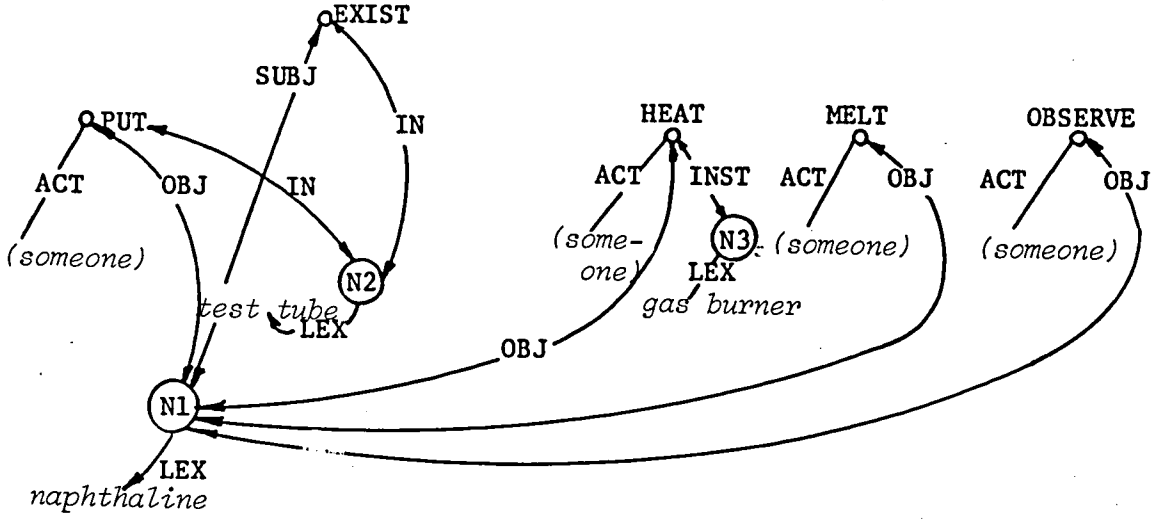


*final result after searching



NS: = ((N1 N3)(N1 N2))

*Though the third and fourth sentences also have empty case makers, they are properly filled in. The following result is obtained.



IV-3-2 Omitted Word in a Noun Phrase

A noun is classified as either an entity word or a relational word. Most nouns have definite meaning by themselves, and are regarded as entity words. However, some kinds of nouns have relational meaning. That is to say, they have slots in their meaning to be filled in by other words, in order that they express definite ideas. Sometimes a relational noun is used alone in a noun phrase. In this case the relational noun must be semantically connected with other words which are omitted in the present noun phrase. Such examples are shown in Fig. IV-3 below.

(1) IOU -O NESSURU TOKI IRO -GA HENKASURU.
sulfur (OBJ, IOBJ) *heat* *when* *color* (SUBJ) *change*

meaning: *When (someone) heats sulfur, the color changes.*

*The phrase 'IRO -GA ' is a noun phrase but it is incom-
color (SUBJ)
plete by itself. We can easily understand the color means 'the color of the sulfur'.

(2) ENSAN -O SHIKENKAN -NI
hydrochloric acid (OBJ) *test tube* (PLACE, TIME, etc.)

20cc IRERU.
put in

meaning: *(Someone) puts 20cc of hydrochloric acid in a test tube.*

*The word 20cc is put in a separate position from ENSAN (hydrochloric acid) in the sentence. It, however, specifies an attribute of the acid, VOLUME.

Fig. IV-3

As the final step in the analysis of a noun phrase, we check whether there remain relational nouns which have no definite meaning. If found, we search through NS for words which are suitable to fill in the slots of the nouns. The searching process is the same as for omitted words in simple sentences. Sometimes the omitted words exist

in succeeding sentences, so we can set up a problem in TL, if we cannot find an appropriate word in the preceding sentences.

IV-3-3 Detailed Description of the Trapping List (TL)

Most anaphoric expressions and omitted words are well analyzed by searching through the preceding sentences. However, we need sometimes to refer to succeeding sentences in order to analyze a sentence properly. The sentences shown in Fig. IV-4 are examples.

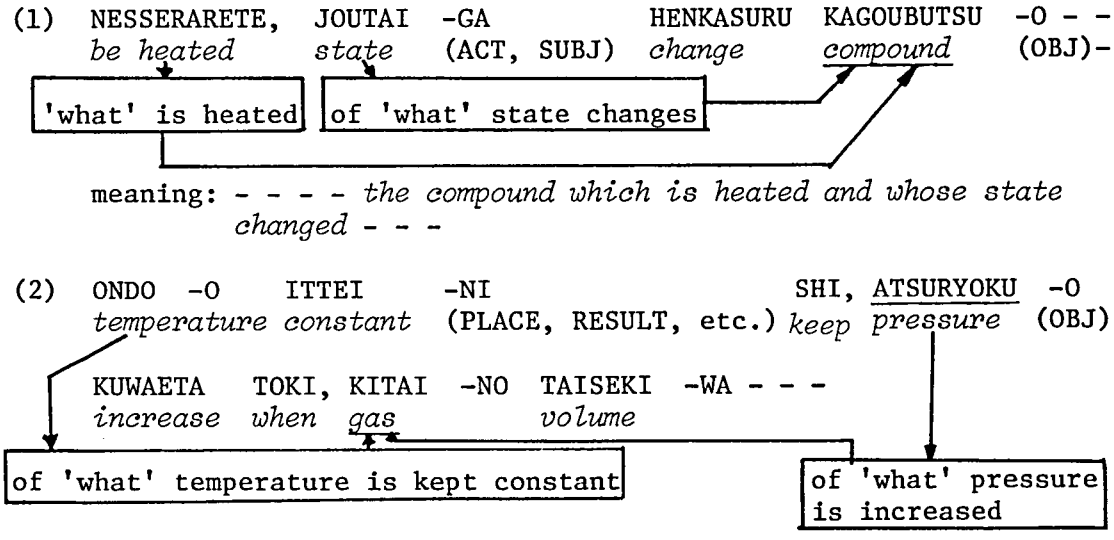
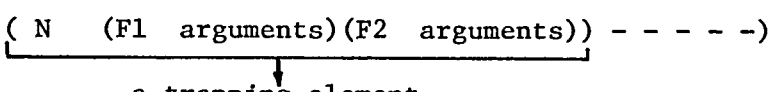


Fig. IV-4 Examples Where Omitted Words Appear in Succeeding Sentences

Because the preceding sentences have already been analyzed and both HNS and NS have been set up, it is easy to refer to the preceding sentences. On the other hand we cannot immediately refer to the succeeding sentences if this is called for.

To solve this problem we set up a trapping list TL. The basic

organization of TL is shown in Fig. IV-5. A trapping element is a



N: number

F1: arbitrary lisp function

F2: arbitrary lisp function

Fig. IV-5 Construction of TL

triplet and corresponds to a pending problem. When we cannot find an appropriate word in the preceding sentences for an omitted word or an anaphoric expression, we put a new trapping element in TL. At this time the first of the triplet, N, is set to zero. When a noun phrase in a succeeding sentence is analyzed, we pick up nouns from the noun phrase one-by-one and check whether the present noun can resolve a pending problem in TL by evaluating the function F1 in the trapping element.

We have defined several LISP functions for the function F1. These functions work as follows

(i) They check whether a noun at hand can solve the problems in TL.

(ii) If it can do so, they update the data (for example, if the function F1 is the function which searches the words in TL for filling in the omitted case element, then the function will put the present noun in the case frame), and return the value 'DELETE'. Then the system will delete the trapping element from TL.

(iii) If it cannot do so, the system adds 1 to N, the first element of the trapping element. When N exceeds five, the trapping element is deleted from TL. That is, it is decided that the problem corresponding to the trapping element can not be solved at all. Before the deletion of a trapping element its third element, the function F2, is evaluated. Thus far F2 has only been used to provide default values to allow some interpretaion for pending problems.

By using the idea of TL, we can separate various checking mechanisms from the main program. They can be invoked automatically when a noun appears in a sentence. The idea of TL resembles that of E. Charniak's 'demon' (1972). When his system encounters a certain word, for example, 'piggy bank', it creates a demon which tries to catch from the succeeding sentences any word (e.g., money) related to the key word. We fear that unnecessary knowledge will clog the system with a 'combinatorial explosion' resulting from the proliferation of demons. Our trapping element is put in TL only temporarily to compensate for any missing elements to be retrieved from succeeding parts. Hence the unnecessary proliferation of elements may be avoided.

IV-4 Processing of Anaphoric Expressions

In Japanese anaphora is expressed by using the articles KONO, KORE, or KORERA which correspond roughly to 'the', 'this' and 'these' in English. The pronoun KORE is used to designate a single object in the preceding sentences, and the pronoun KORERA is used to designate plural objects. The article KONO is used as a constituent of a noun phrase. Though the articles in English modify the first succeeding noun, KONO often modifies a noun at some distance. An example is given in Fig. IV-6.

	noun		noun		noun
KONO	SHIKENKAN	-NO	NAKA	-NO	DOU
this	*test tube*		*in*		*copper*

{ *the copper in this (inside of) the test tube*
{ *this copper in the test tube*

Fig. IV-6 Example of the Article KONO which Modifies a Noun at Some Distance

In this example there are three nouns following the article which can be modified by it syntactically. We must decide the preferable modification pattern by using contextual information. In the analysis of a noun phrase, we scan the words one-by-one from left to right. When we catch the article KONO, we put it in the temporary stack. The word will then be checked to see whether it can modify a noun in the following noun phrase. When we scan the noun SHIKENKAN (test tube) in Fig. IV-6, we check whether the object indicated by it was already mentioned in the preceding sentences. If it was, then the article KONO is regarded as modifying the noun 'test tube'. If not, the article is stacked again. In this way the article will be checked against the nouns in the noun phrase until the noun modified by it is found.

The article KONO is used in the following two ways:

(1) SANSO -GA ARU. KONO SANSO -O - - - -
oxygen (SUBJ, ACT) *exist* *oxygen* (OBJ)
There is oxygen. *The oxygen - - - -*

The noun SANSO modified by the article KONO is the same entity noun which appears in the first sentence.

(2) SANSO -GA ARU. KONO TAISEKI -O
oxygen *exist* *volume* (OBJ)
There is oxygen. *The volume of the oxygen - - - -*

In this case KONO alone designates the entity noun SANSO which appears in the first sentence. This usage is permitted only if the noun modified is a relational noun. If the noun has only a relational meaning, the second usage appears more often than the first.

The meaning descriptions of articles and pronouns like KONO are procedually expressed by LISP functions. The functions in the dictionary will be evaluated if we find such words in a sentence. The function for KONO operates in the following way.

Step 1. A check is made to see if the succeeding noun is relational. If the noun has only a relational meaning, it is first assumed that the article KONO is of the second usage and we go to Step 3.

If not, we go to Step 2.

Step 2. The first usage of KONO has the following three varieties.

(i) SANSO -GA ARU. KONO SANSO -O - - - -
There is oxygen. *The oxygen* - - - -

The noun modified by the article is the same noun which appears in the preceding sentence.

(ii) SANSO -GA ARU. KONO KITAI -O
There is oxygen *The gas* - - - -

The noun '*gas*' modified by the article is an upper concept noun of the referent noun '*oxygen*'.

(iii) SANSO -TO SUISO -O KONGOUSURU. KONO KONGOUKITAI
oxygen and hydrogen (OBJ) *mix* *The gas mixture*
-O - - -
(OBJ)
(Someone) mixes oxygen and hydrogen. *The gas mixture*
- - -

The article modified a nominalized form of the fitst sentence. The first sentence instantiates the case frame of the verb '*mix*'. We evaluate the NTRANS description of the case frame and obtain a new inferenced object '*mixture*', whose elements are the oxygen and the hydrogen. The noun KONGOUKITAI modified by the article is a lower concept noun of the inferenced noun (mixture) in HNS.

According to these three varieties, we provide the following three check routines. The order of checking is shown in Fig. IV-7.

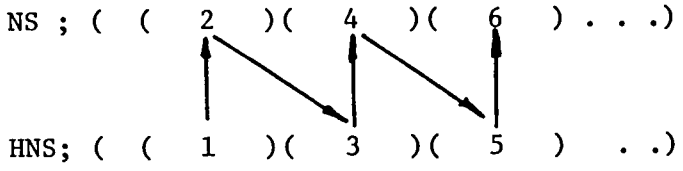


Fig. IV-7 The Order of Checking

(check 1) Is there in the list the same noun as the noun modified by KONO.

(check 2) Is there in the list a lower concept noun of the noun modified by KONO.

(check 3) Is there in the HNS list an upper concept noun of the modified noun, and are its properties consistent with those of the modified noun.

If we can find a noun which satisfies one of these three conditions, we decide that it is the referent noun. If we cannot, the function for KONO returns the value NIL.

Step 3. If the noun which follows the article has a relational meaning, the meaning description of the noun has slots which must be filled in by other words. What kind of noun is preferable for a slot is described in the meaning description. We search in NS and HNS for an object which satisfies the description. For example suppose the input is

SANSO	-GA	ARU.	KONO	TAISEKI - - -
oxygen	(ACT SUBJ)	*exist*		*volume*

The noun TAISEKI is an attribute noun. So we look for a noun which may have the attribute and recognize that oxygen is appropriate. Another example is

SHIKENKAN	-GA	ARU.	KONO	NAKA	-NI - - -
test tube	(ACT SUBJ)	*exist*		*in*	(PLACE, RESULT)
There is a test tube.			*In the (test tube) - - -*		

The noun NAKA (in) is a prepositional noun which requires a '*container*' or '*liquid*'. We can easily recongnize the test tube as a lower concept noun of '*container*'. Therefore we assume the word KONO is used for the test tube. If we find no such nouns, we suppose that the article KONO is not of the second usage but of the first. So we will go to Step 2.

The pronoun KORE (this, it) is used in sentences as a case element. We can predict the kind of objects designated by the

pronoun by using the case frame description of the verb in a sentence. The postposition attached to the pronoun indicates a set of possible cases. By taking from the frames the cases which belong to the set, we can obtain the semantic descriptions which are satisfied by the object designated by the pronoun. So we search through HNS and NS for an object which satisfies the descriptions. Consider the following:

MIZU	500cc	-GA	ARU.	KORE	-NI	SHOKUEN
water		(ACT, SUBJ)	*exist*		(PLACE, RESULT, TIME - - -)	*salt*

2gr	-O	IRERU
	(OBJ)	*put in*

There are 500cc of water. *In this (water) (someone) puts in 2 grams of salt.*

The set of possible cases for the postposition NI is (PLACE, RESULT, TIME, BENEFICENT - - -), and the case frames of IRERU (*put in*) have the case PLACE. We can predict that the pronoun KORE (*this, it*) fills the PLACE case in the sentence. The semantic description says that a lower concept noun of '*container*' or '*liquid*' is preferable as the PLACE case of the verb IRERU *(put in)*. The object '*water*', which is a lower concept noun of '*liquid*', is found in NS, and is determined to be the object designated by the pronoun.

We have some other pronouns and articles in Japnese which are analyzed in the same way. We provide different LISP functions for different pronouns and put them in the dictionary definitions of these words.

T. Winograd treated the same problems in his excellent system SHRDLU (1972). However, the world which his system can deal with is very limited. In order to construct a system which can treat a wider range of sentences, the system should be equipped with the schema representing the relationships between events and objects (an event may imply the occurrence of new objects or changes in the

properties of objects). In real world sentences, there exists more complex phenomena about anaphoric expressions and omissions of words than those treated in SHRDLU. We do not claim that our system can treat such complex phenomena, but we hope that our system can be evolved to cover such phenomena by means of combining contextual analysis procedure with semantic descriptions of words.

IV-5 Analysis of Complex Sentences

In the previous sections we described the semantic and contextual analysis procedure of our system. In this section we explicate by using example sentences how these functional units are organized in order to analyze fairly complex sentences.

Suppose the input sentence is

ASSHUKU-SARETE	TAISEKI	-GA		HENKA-SURU	TOKI	-NO	SANSO
be compressed	*volume*	(SUBJ, ACT)		*change*	*time when*		*oxygen*

-NO	JOUTAI	-O	KANSATSUSHI,	SONO	ATSURYOKU	-O	SOKUTEISHI,	
	state	(OBJ)	*observe*	*the*	*pressure*	(OBJ)	*measure*	

SORE	-O	GURAFU	-NI	ARAWASU.
it	(OBJ)	*graph*	(PLACE, RESULT)	*express*

(Someone) observes the state of the oxygen when it is compressed and the volume (of it) changes, measures the pressure, and expresses it by a graph.

The sentence is analyzed by the following steps.

1. The program first tries to find the leftmost verb, and analyzes the clause governed by the verb. The sentence ASSHUKUSARETE (*be compressed*) is analyzed first. This sentence has an irregular structure in the sense that there are no explicit case elements before the verb. All case elements are omitted in this sentential

part. By checking the inflection of the verb (ASSHUKU-SURU (*to compress*) --- ASSHUKUSARE (*to be compressed*)), we recognize that the sentence is in the passive voice. The lexical description of the verb in the word dictionary indicates that it takes two intrinsic cases, that is ACTOR and OBJECT. In a Japanese sentence especially in the field of chemistry, the case element ACTOR is apt to be neglected. Therefore we adopt a dummy filler for the ACTOR to represent the author of the sentence or some other human being. As there are no preceding sentences, we cannot fill in the OBJECT case immediately. So we set up the pending problem in TL which watch the analysis of the succeeding strings to fill the gap.

2. The clause TAISEKI-GA HENKA-SURU will be analyzed next. The verb HENKA-SURU (*change*) requires only SUBJ case. The postposition GA attached to the noun TAISEKI (*volume*) possibly implies the case SUBJ. The noun TAISEKI is a lower concept noun of 'attribute', which satisfies the semantic condition for the case element. So this sentence is analyzed in a straightforward manner. However, because the noun TAISEKI is an attribute noun, we must find the corresponding entity noun. That is, we must identify the object whose volume is being referred to. As we cannot find such an object in the preceding sentences, we set up a pending problem in TL. By checking the inflection of the verb HENKASURU (*change*) and noting that it is immediately followed by a noun, it is recognized that the sentence is an embedded sentence modifying the following noun TOKI (*time, when*). We then connect this sentential part with the noun TOKI by using the relation SMOD (MODified by a Sentence).

3. When we analyze the next clause,

TOKI	-NO	SANSO	-NO	JOUTAI	-O	KANSATSU-SURU
time		*oxygen*		*state*	(OBJ)	*observe*
when						

We first perform the analysis of the noun phrase TOKI-NO SANSO-NO JOUTAI. The combination of the two nouns TOKI (*time*) and SANSO

(*oxygen*) is semantically permissible because '*oxygen*' is a lower concept noun of '*material*', and can be modified by a word which designates a special point of time. The noun TOKI (*time*) is modified by the sentential part analyzed at Step 2, and designates the time when the event expressed by the sentential part occurs. The combination of SANSO (*oxygen*) and JOUTAI (*state*) is also permissible.

The nouns TOKI (*time*), SANSO (*oxygen*) and JOUTAI (*state*) in the noun phrase activate the trapping elements in TL. The noun SANSO (*oxygen*) satisfies the conditions of the two trapping elements set up by Step 1 and 2. That is, SANSO (*oxygen*) fills in the case OBJ of the first clause. TAISEKI (*volume*) in the second clause is regarded as the volume of the oxygen in the current clause.

4. The next clause ATSURYOKU-O SOKUTEISHI presents no new problems. However a referent for the noun ATSURYOKU (*pressure*) must be found. '*oxygen*' in the preceding sentence is easily found to satisfy the conditions for having the quality ATSURYOKU (*pressure*).

5. The remaining steps follow along similar lines. The results of the parsing of the expression are shown in Fig. IV-8.

The next example shows how HNS is used. Suppose the input sentence is

(1) Input sentence:

ASSHUKUSARETE, TAISEKI -GA HENKASURU TOKI -NO
be compressed *volume* (ACT SUBJ) *change* *when*

SANSO -NO JOUTAI -O KANSATSUSHI, SONO ATSURYOKU -O
oxygen *state* (OBJ) *observe* *the pressure* (OBJ)

SOKUTEISHI, SORE -O GURAFU -NI SURU.
measure *it* (OBJ) *graph* (IOBJ, RESULT, etc.) *represent*

meaning: *(Someone) observes the state of oxygen which is compressed and whose volume changes. (Someone) measures the pressure and represents it as a graph.*

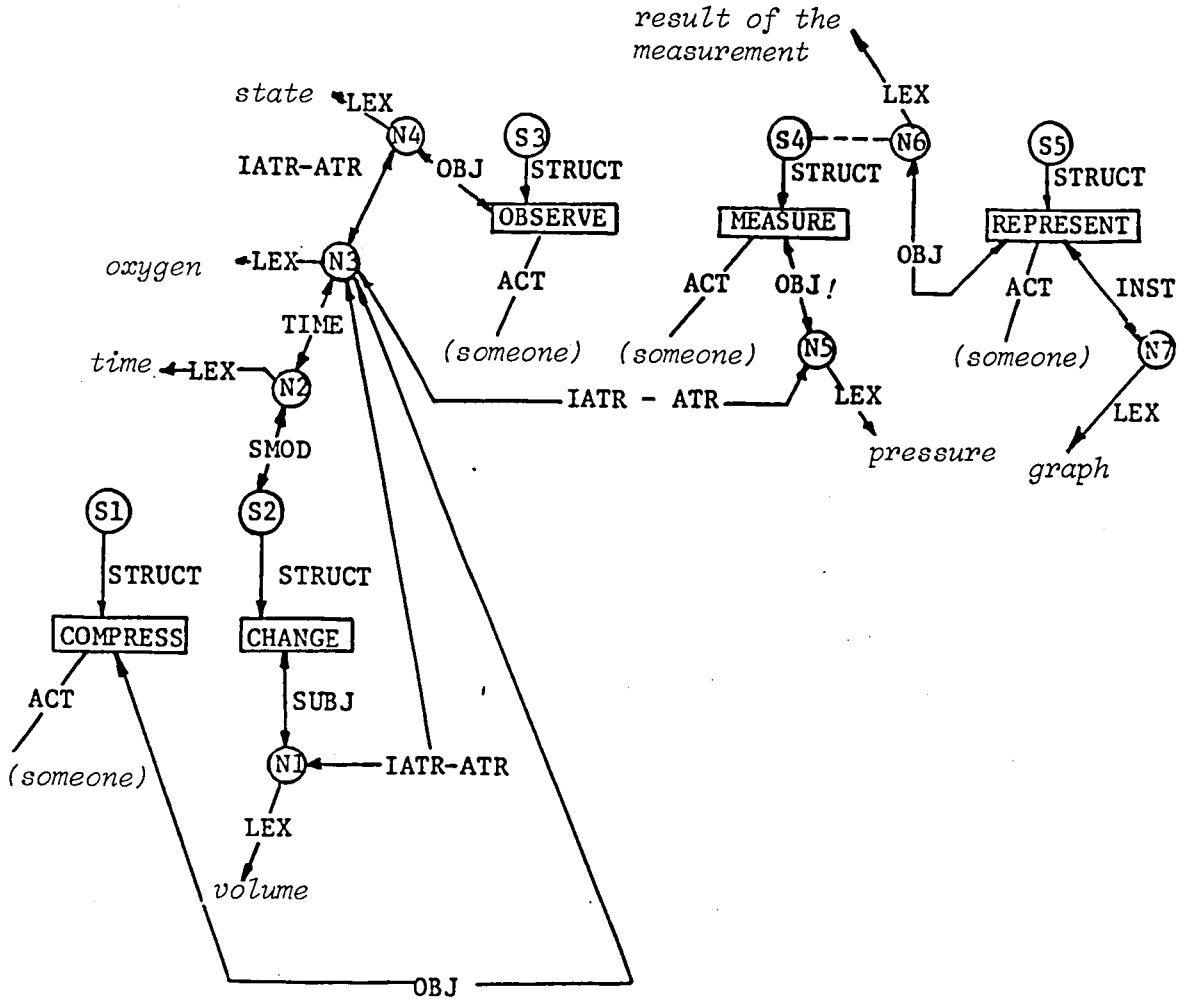


Fig. IV-8 Graphical Representation of the Analysis Result

SUISO	-TO	SANSO	-O	KONGOUSHI,	KONO	KONGOUKITAI	-NI	
hydrogen	*and*	*oxygen*	(OBJ)	*mix*		*the*	*gas mixture*	

TENKASURU	-TO	BAKUHATSU-SHI,	MIZU	-GA		DEKIRU.
fire	(*if, when*)	*explode*	*water*	(SUBJ,	ACT)	*be made*

If (someone) mixes hydrogen and oxygen, and fires the gas mixture, then (it) explodes and water results.

The following steps are performed.

1. When the analysis of the first clause SUISO-TO SANSO-O KONGOUSHI is complete, the case frames of the verb KONGOUSHI are instantiated. The NTRANS expression of the case frame which obtains the highest matched value is determined. As the result a new object '*mixture*' is created and the elements of the mixture are hydrogen and oxygen. This newly created object is put into HNS.

2. The noun phrase KONO KONGOUKITAI-NI (*to the gas mixture*) in the clause is modified by the anaphoric determiner KONO (*this*) which requires a referent. The noun KONGOUKITAI (*gas mixture*) is a lower concept noun of '*mixture*' having as components gaseous objects. We search in the HNS and HS and find the object '*mixture*' in HNS whose elements are the hydrogen and the oxygen.

3. The object '*gas mixture*' is the theme of the succeeding sentences. It fills in the omitted case ACT of the third clause and FROM case of the fourth clause. Fig. IV-9 shows the result of the parsing.

Table IV-2 below shows the score obtained by applying our parsing program to the sentences in a junior high school chemistry textbook.

(2) Input sentence:

SUISO	-TO			SANSO	-O	KONGOUSHI	
hydrogen	(conjunctive pp - - - and)			*oxygen*	(OBJ)	*mix*	
KONO	KONGOUKITAI	-NI			TENKASURU	-TO	
this	*gas mixture*	(OBJ, IOBJ, PLACE, etc.)			*ignite*	(conjunctive pp - - - *if, when*)	
HAGESHIKU	KAGAKUHENKASHI,	MIZU	-GA		DEKIRU.		
violently	*react*	*water*	(ACT SUBJ)		*be produced*		

meaning: *If (someone) mixes hydrogen and oxygen and ignites it, then the gas reacts violently and water is produced.*

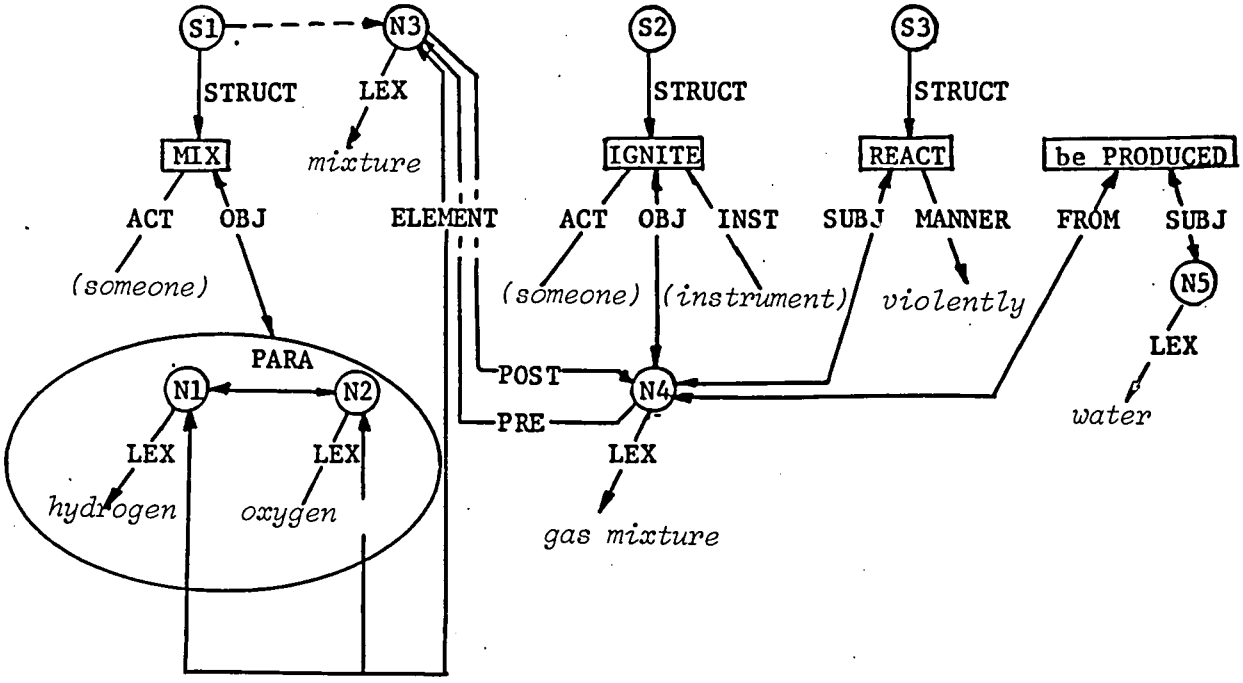


Fig. IV-9 Graphical Representation of the Analysis Result

	total no. tried	successes	failures
noun phrases	312	286	26
conjunctive phrases	372	349	23
sentences	280	254	26

Table IV-2 Successes and Failures Schema

IV-6 Conclusion

We can summarize our interpretive procedure as follows:

(1) Through the use of grammatical case we describe patterns of

activity in the verb dictionary. The descriptions also contain information as to how activities are connected with each other and how activities change objects.

(2) The meaning descriptions of nouns are based upon the upper and lower concept relationships and attribute value pairs. Some kinds of nouns are regarded as having relational meanings. Their meaning descriptions are similar to those of verbs, adjectives, and prepositions. By using these descriptions we can analyze fairly complex and long noun phrases where there are few syntactic clues.

(3) We do not use logical expressions to represent context. Contextual information is represented in the form of what we call intermediate term memory. This in combination with the semantic descriptions of words has enabled us to perform efficient analyses dependent on contextual information.

(4) We have developed a programming language which makes it easy to write grammars for natural language and to control the analysis procedure. By using this language, we can incorporate naturally semantic and contextual analyses into syntactic analysis. We do not need a large and involved program which is responsible for the semantic interpretation of the output given by the syntactic analysis component. Instead, we provide many simple and small functions for semantic and contextual analyses.

We have obtained fairly good results with our approach. The contextual analysis program on the other hand can treat only local contexts. In order to treat more global contexts, we feel the following improvements will be necessary.

(i) We must provide our system with an appropriate schema corresponding to human long term memory in order to represent the state of the world. The system must have frameworks to express spatial relationships among objects, time relationships among events and so on.

(ii) At the present stage we have only one relationship CON

to connect one activity with another. However, human knowledge of the world accommodates various kinds of relationships among activities, such as cause, purpose, reason, etc. These relationships may play an important role not only in the analysis of sentences, but also in the inference processes in answering a question.

(iii) The descriptions of verb meanings using case work rather well for analyzing verb-centered sentences. However, the results of analysis depend on what verbs are used in surface sentences. Hence, the sentences which convey the same meaning but are expressed by using different verbs may be transformed into different internal representations. This is a serious drawback when constructing question-answering systems or other kinds of intelligent systems.

(iv) In order that a system be able to communicate with people in a flexible and natural manner, it must be able to derive inferences from incomplete data bases. Therefore we must design a procedure other than the uniform proof procedure such as the resolution proof procedure.

(v) It is necessary to apply our method in fields different from chemistry and to test whether our semantic description method should be changed or not.

There are many scholars who are interested in using case structures as a representation of natural language utterances. B. Bruce (1975) offers a good survey and a unified point of view in favor of case systems. We also believe that the case system is a promising approach to the representation of meanings in natural language.

120項欠

CHAPTER V

SEMANTIC NETWORK AS INTERNAL KNOWLEDGE REPRESENTATION

V-I Introduction

Many researchers are convinced that various kinds of background knowledge which are not linguistic, play important roles in the process of language understanding. As we chose elementary chemistry to be the micro world of our question-answering system, we should provide our system with real-world knowledge of this field. In Chapter III and Chapter IV, we explained the forms of description used in the Noun Dictionary and the Verb Dictionary of this micro world, and also described how they are utilized in order to disambiguate the analysis of sentences. The meaning descriptions in the dictionary reflect the possible relationships in the real world between objects and objects (in the Noun Dictionary) or objects and events (in the Verb Dictionary). They are, however, organized in a form which is convenient for being utilized during the analysis of sentences. So the descriptions contain much information which is irrelevant for doing deductions or problem solving and, moreover, lack necessary information for doing deductions and problem slovings. An appropriate framework of representing knowledge must be devised for such purposes.

In this chapter, we will introduce the idea of the semantic network as a representation scheme for use in deduction and problem solvings. Then we will give a specification of our semantic network, and show how to implement and use it in the problem-solving processes.

V-2 Internal Knowledge Representation -- Survey of Related Research Works

The term *semantic network* means different things to different people, because the definitions of semantic network given so far have been vague. Many different kinds of representation schemes are called by the name 'semantic network' and there are also many representation schemes called by names other than semantic network which share the same basic ideas of our system. In the following, we will describe some of them in comparison with our semantic network.

1. Quillian's semantic network : R.Quillian is the first man who proposed the semantic network as an internal knowledge representation. His program (TLC - Teachable Language Comprehender) can deduce the relationships between two words in a phrase by traversing the net to find the *visual* immediacy of *interrelationships* between concepts denoted by the words. For example, from a phrase like 'lawyer's client', the program can derive the following explanations.

'The client who employs the lawyer who gives advice about legal matters to the client'

His program is based on the notion of *associativity* between concepts. Associativity between any pair of concepts is represented by a link. In his system, both soundness and completeness of the system in the logical sense are ignored. The conclusion or the explanation derived by his system is not always logically correct.

2. Raphael's SIR : Raphael did not ever claim that his representation was a semantic network. However, the structure in which his system represents information shares many features in common with the currently developed networks. His main interests are how to represent logical relationships such as set-membership, part-whole, ownership and so on, and how to utilize these relationships to deduce a

conclusion from given facts. The conclusions deduced by his system are guaranteed to be logically correct. Fig. V-1 shows some examples of tasks which were performed by his system.

(... Every boy is a person)

(I understand)

(... A finger is part of a hand)

(I understand)

(... Each person has two hands)

(The above sentence is ambiguous .. Please re-phrase it)

(... There are two hands on each person)

(I understand)

(... How many fingers does John have Q)

(I don't know whether finger is part of John)

(... John is a boy)

(... Every hand has five fingers)

(... How many fingers does John have Q)

(The answer is 10)

Fig. V-1 Sample Dialogue of SIR

The above two systems are typical of early approaches to the semantic network as scheme for representing knowledge. The semantic network approach has the following two notable characteristics:

1. *Retrieving relevant information by traversing the network*. In this context, links in the network express mainly the associative relationships among concepts.

2. *Performing logical deduction or problem solving by traversing the network*. In this context, as compared with the first one, links in the network are regarded as expressing the logical relationships among concepts.

Recent research efforts have attempted to make clear the properties of the semantic network in the second context. R.F. Simmons (1977), L.K.Schubert (1976), G.G.Hendrix (1975a, 1975b, 1977), J.Mylopoulos (1975, 1977), J.Minker (1977) and others are interested in the expressive power of semantic networks. They examined the logical properties of conventional semantic networks, developed a more rigid representation format, and some of them formulated the logically adequate operations on the network. Some people, especially psychologists like D.E.Rumelhart and D.A.Norman, are interested in the associative property of semantic networks (Norman, 1975). These approaches are very interesting, but from the point of view we now occupy, we will exclude the psychological topics from the current discussion because it is out of our interests in this thesis.

Before discussing the network we have recently developed, we will first examine the advantages and disadvantages of semantic networks by considering more primitive network systems.

SIR developed by B.Raphael (Raphael 1968) and the kinship system (SAD-SAM) by L.Lindsay are typical of these primitive systems. Lindsay's system is a highly specific one oriented toward kinship relations. This point illustrates both the advantage and disadvantage of semantic networks. One can construct an efficient problem solving or deduction system for a specific domain at the sacrifice of generality. The program interpreting the network gives the semantics of the representation. That is, what the nodes and links in the network really represent are determined by the program which utilizes the network. In the early stage of network systems, the designers constructed the program in forms convenient for their specific purposes, so that the semantic networks in different systems have different semantics. This is the reason why the term *semantic network* has had different meanings for different people. Though such kinds of systems are efficient for their own purposes, for different purposes one have to design completely different formats for networks and the

procedures that interpret them. Lindsay's kinship system is typical, in the sense that his system shows remarkable performance but can only be applied to a very limited field, i.e. kinship relations.

The purpose of Raphael's system was to develop a representational format for more general knowledge. He declared his attitude in his paper (Raphael 1968) as follows.

'The SIR system is based on a single model which captures some of the advantages of various specific models while permitting uniform procedures as well as the storage and retrieval of arbitrary facts ...'

In his system, all the factual knowledges represented in relational statements and they are organized in a kind of network structure. A link in the network expresses a certain kind of relationship between objects and classes of objects, such as ownership, part-whole relationship, left-right spatial relationship, set-inclusion relationship,or set-membership. The links in his system are, therefore, considered as *predicates*. Because every predicate in his system must be expressed by a link, one cannot express predicates which have more than two arguments. In this sense, his representational format is restricted. However, this restriction enables his system to have an interesting characteristic, namely that both retrieving relevant knowledge and performing logical deductions are done at the same time by traversing the network. A more serious disadvantage of his system follows the fact that although each factual datum is represented by a certain network structure, the axiomatic information for each relation is expressed in the form of program. Each separate information storage or retrieval operation for each different relation is controlled by a different subprogram. Because the axioms about a relation usually bear reference to other relations, each subprogram is highly dependent on other subprograms. Moreover, if the user wants to define a new relation, he must not only write a new subprogram for the relation but also modify the

other related subprograms. As mentioned before, in the early network systems the meaning of links and nodes, that is, the semantics of networks were given by the programs which interpret the networks. Raphael cleverly noticed this fact. His system can be changed to be applicable for any problem domains by adding subprograms for the relations which are useful in that domain. His system is general in this sense. But this means only that one can theoretically write a program to do whatever one wants. The important consideration is what kinds of useful mechanisms the framework provides the programmer, and how easily the programmer can express what he wants in this framework. The SIR system has very few provisions for programming the subprograms. It is also an important factor whether the system provides useful mechanisms for extension and enhancement. The SIR system has very few provisions for adding new subprograms.

Any discussion concerning a general framework for representing knowledge would be incomplete without a mention of predicate calculus and recently developed AI languages, though they are not semantic networks. Predicate calculus seems to give the most complete and rigorous framework for representing knowledge. In particular, the first order predicate calculus was used by some researchers as the internal representation of knowledge in their question-answering systems, because an efficient proof procedure for this kind of logic has been developed. The procedure is complete in the sense that if a theorem is logically deduced from the given axiom, the procedure can eventually prove it. However, the procedure has the following disadvantages when applied to a question-answering system.

1. It is obvious that the procedure should have the ability to select the facts (axioms) relevant to the given question because there may be a great number of facts which are not relevant to the current problem. We believe that, in the computer memory, similar facts should be grouped together. However, the proof procedure itself has no such ability to cluster relevant knowledge.

2. In order to apply the proof procedure to a question-answering system, all kinds of information must be expressed as predicate calculus formulas. People, however, use various kinds of knowledge other than those that can be expressed in logical formulas. For example, we know not only a theorem itself but also how to use it, when the theorem is useful, and so on. These kinds of second order knowledge should be reflected in the control of the deduction or problem solving processes.

3. Finally, in a question-answering system, we must accept natural language sentences as input. Then we have to translate them into formulas of the first order logic. In the analysis and translation of input sentences, we must refer to general knowledge. If we represent general knowledge in logical formulas, it must be logically consistent. Usually our knowledge is locally consistent but very often violates consistency globally. So it is very hard to keep the entire knowledge base always consistent.

Because the last problem was discussed in the previous chapters, we will express here our attitude to the first two problems. In the SIR system, one can embed arbitrary heuristics in the subprograms for interpreting the links, because the system provides no strong framework for programming. As a setback, it has the disadvantage that one must prepare by himself all the implications of the representation which he chooses. On the contrary, the system based on the predicate calculus formulas and the proof procedure imposes strong constraints on the programmer as to how to encode his knowledge. Because the system provides a general procedure for interpreting the expressions, the semantics of the expressions in which the programmer encodes his knowledge are completely specified by the system. Moreover, the general proof procedure can theoretically deduce whatever logically follows the given facts. One does not have to anticipate how the given facts will be used. A certain facts can be used in the way which the designer never anticipates.

On the other hand, because the way to select releyant facts from the set of given facts are completely left to the general proof procedure, many irrelevant proof paths are generated during the proof process. Though several refinements of the general proof procedure to suppress the generation of irrelevant proof paths have been proposed, these refinements are not satisfactory. They are based on only the surface forms of the formulas. However, the predicate calculus formula imposes such a rigid and limited representational framework that various useful types of knowledge, especially the knowledge about how to control the proof process, cannot be expressed. That is the reason why such refinements are not so successful (C.L.Chang 1973).

Procedural approaches which are popular nowadays are frameworks which make it easy for the designer to directly control which facts will be used when. The procedural approaches exhibit remarkable performances in a certain restricted domain. However, the approaches have more or less the same disadvantage that the early representation schemes had. Though AI languages prepare various kinds of useful programming tools such as automatic backtracking in PLANNER, a context mechanism in CONNIVER and so on. The frameworks which such languages provide are much richer than those in early systems, but the designer of a system must still provide too much advice for interpreting his expressions. The *pattern directed invocation of a procedure* is one of the commonly used techniques in the AI languages. By virtue of this technique, the appropriate procedures are invoked at appropriate times without writing any explicit calling processes. Therefore, one can evolve his system simply by adding new procedures with the patterns which describe when they are needed. However, because the method of representing the purposes of procedures by using patterns are completely dependent on the designer, the semantics of patterns are not determined by themselves. The semantics of patterns are determined only by the

procedures which are invoked by those patterns. It is often the case that two different patterns convey the same meaning in a certain context and therefore they can be applicable to a certain kind of problem, or conversely that two procedures with the same invocation pattern have different implications in a certain context.

Though in these AI languages the patterns are organized by some index structures which are convenient for retrieving them, the index structures are constructed only by considering the syntactic structures of patterns so that the programming system is not sensitive to subtle implications of the pattern's meaning. In other words, the designer should still pay attention to the implications of the patterns which he uses (D.G.Bobrow 1974).

We think that the framework must provide much richer index structures which reflect the semantic meaning of patterns and the framework must also provide some general abilities. By the word 'general abilities' we mean the abilities which are universally applicable to any domain, that interprets and manipulates the patterns. Throughout this discussion, we used the term 'pattern' to indicate some descriptions attached to a procedure. The 'pattern' in the present AI languages means a simple list structure that has no meaning by itself. It plays only the role of a handle for retrieving procedures. However, our contention is that a much richer descriptive framework is necessary for more complicated systems. It is also desirable that the descriptive expressions convey some meanings of their own. Our semantic network can be seen as a certain semantical index structure for various kinds of knowledge domain whether it be procedural or descriptive. By traversing the network, the system not only retrieves relevant knowledge but also performs certain kind of deductive operations at the same time.

Though the resolution-based general proof procedure is inefficient, many logical operations can be well explained and formulated by using the predicate calculus. We will examine by using predicate calculus formulas, what traversing our network logically

means and whether a certain operation on the network is logically valid. In other words, we will examine the correspondence between the notation of predicate calculus and our semantic network representation.

Recent research efforts on semantic networks have attempted to clarify the logical properties of semantic networks and to extend the expressive power of networks in order to cover the expressive power of first order or higher order predicate calculus. We are also interested in examining the parallelism between the operations on the network and predicate calculus. The main purpose of the notations which have been developed by G.Hendrix, L.Schubert and so on is to extend the expressive power of networks so that their network notations can be interpreted as logical formulas. In a sense, their networks are the graphical notations of logical formulas. Hendrix developed a proof procedure by using his representations, but the procedure does not utilize the advantage which network representations have in general. The advantage of network representations, we think, is that both the selection of relevant knowledge and the execution of logical operations are carried out at the same time simply by traversing networks. We are much more interested in constructing a network which has the form convenient for this purpose. We are presently not so much interested in constructing a network which has the same expressive and deductive power as the predicate calculus. The network which we have developed gives the framework in which various kinds of knowledge, such as procedural knowledge, knowledge about external data bases, and so on, can be naturally embedded. The validity of the format of the network representations and the validity of the operations on the network will be examined by logical expressions. A programming language by which we can define the network will be also given.

V-3 Definition of the Semantic Network - S.N.

In the following, the semantic network which we define will be called S.N. to distinguish it from the other kinds of semantic network. We are going to define the S.N. in comparison with ordinary predicate logic. We have the following basic node types in the S.N.:

1. *Predicate Node*
2. *Function Node*
3. *Variable Node*
4. *Constant Node*
5. *Logical Connective Node*

These node types directly correspond to the predicate, function, individual variable, individual constant, and logical connective in predicate calculus, respectively. In order to embed in the S.N. additional information which cannot be expressed naturally in a predicate calculus formula, we will further sub-divide them according to the functions which the nodes perform.

V-3-1 Variable Node and Constant Node

In the first order predicate calculus, we have a unique set of elements which is called Domain. All the constants and variables are assumed to be elements in the domain. However, there are usually more than one domain as in many-sorted logic (Sandwall 1970). In the present system, we have the following three domains.

1. *Domain of Materials*
2. *Domain of Markers*
3. *Domain of Numerals*

A variable or a constant node in the S.N. belongs to one of these three domains. The graphical notation of a variable and a constant node is given in Fig. V-2.

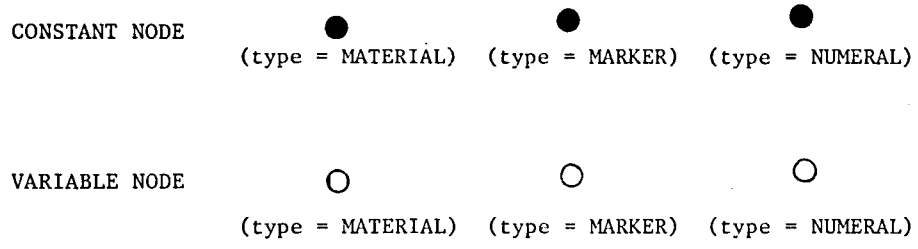


Fig. V-2 Graphical Notation for Constant Nodes and Variable Nodes

V-3-2 Function Node

A function defines a mapping that maps a list of constants to a constant. In the S.N., we will subdivide the function node into the following two types according to the type of the value of the function.

1. *Function Node* (type = *Material*) : The value of a function of this type is the element which belongs to the domain *Material*.
2. *Function Node* (type = *Attribute*) : The value of a function of this type is an element which is a marker or which belongs to the domain *Numeral*.

The following links are attached to a function node.

1. Argument link : notated in the S.N. as ARG1, ARG2, ...
2. Value link : notated in the S.N. as VALUE.

A graphical notation of a function node is shown in Fig. V-3. Note that in a graphical notation every link is represented as a unidirectional link but in a real implementation a link was represen-

ed as a bi-derectional.

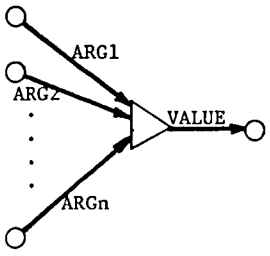


Fig. V-3 Graphical Notation for a Function Node

V-3-3 Predicate Node

Predicate symbols are the most important type of symbols in the first order predicate calculus. Various kinds of information are expressed by using predicates. For example, the expressions LOVE(x, y), HUMAN(x), and ON(x, y) in the ordinary predicate calculus are written in the same format using predicate symbols. Therefore, all the expressions are stored in the same storage structure and used in the same manner during the deduction process. This is one of the reasons why the universal proof procedure is inefficient when it is applied to the real world problems, namely that it may attempt to use any fact at all in constructing a proof.

In the S.N., we classify the predicates into the following five types and provide a different storage structure and a different proof procedure to each different type.

1. *Concept predicate ... Concept Node*
2. *Arithmetic predicate ... Arithmetic Node*
3. *Relation predicate ... Relation Node*
4. *Boolean predicate ... Boolean Node*
5. *Event predicate ... Event Node*

All these predicate nodes, however, are expressed in the same graphical notation. The graphical notation for predicate nodes is given in Fig. V-4.

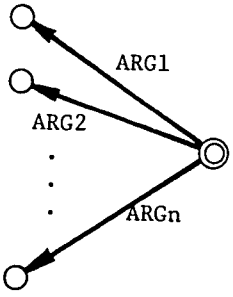


Fig. V-4 Graphical Notation for a Predicate Node

V-3-3-1 Concept Node

A concept predicate is a one-place predicate and specifies a certain set in some domain. For example, the predicate 'chemical-compound' specifies the set in the domain MATERIAL which consists of the elements being chemical compounds.

According to the notation of S.Oshuga (Oshuga 1977), the formula of the ordinary first order predicate logic

$$(\forall x)(\forall y)[A(x) \wedge B(y) \longrightarrow P(x, y)]$$

is expressed by

$$(\forall x/\mathbb{A})(\forall y/\mathbb{B})P(x, y),$$

In this notation, the ranges of values that the variable x and y can take are restricted to the sets $\mathbb{A}$ and $\mathbb{B}$ which are specified by the predicate A and B, respectively. To make the reasoning process efficient by introducing a hierarchical relationship among the sets is one of the frequently used techniques in semantic networks and other recently developed systems. Concept nodes in the S.N. are used to realize this basic mode of reasoning(see section V-4).

Examples of concept nodes are shown in Fig. V-5.

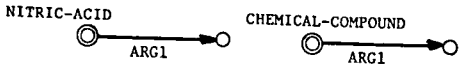


Fig. V-5 Example of a Concept Node

V-3-3-2 Arithmetic Node

As pointed out by G.Hendrix(Hendrix 1977), for many applications it is important for the system's knowledge base to include sources of information such as relational data bases or arithmetic algorithms which are external to the network. Some types of predicates in the first order predicate calculus are used to describe a certain dependency relationship between elements in domains. By the term 'dependency relationship' we mean a relationship like 'functional dependency' in the theory of relational data bases. An arithmetic predicate is one of such types of predicate.

An arithmetic predicate(node) shows that there exists a certain computable relationship between the arguments. We have the following arithmetic nodes in the S.N. at present.

1. DIVide/MuLTiply
2. ADD/SUBtract
3. EXPonential

The fact

'if x is a solution, then there exists the ADD/SUB relationship between the mass of x, the mass of the solute of x, and the mass of the solvent of x'

is represented in the S.N. as shown in Fig. V-6. Note that this representation can be interpreted as the following ways.

'if x is a solution and both the mass of the solute of x and the mass of the solvent of x are known, then you can compute the mass of x by adding them.'

'if x is a solution and both the mass of x and the mass of the solvent of x are known, then you can compute the mass of the solute of x by subtracting them.'

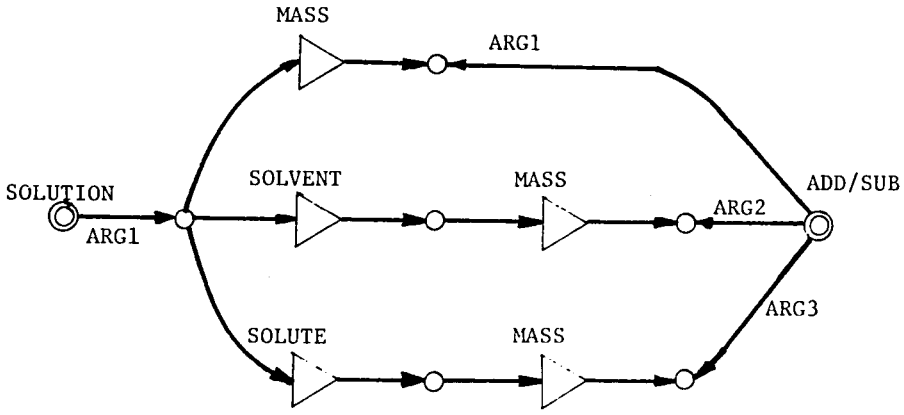


Fig. V-6 Example of Arithmetic Nodes

V-3-3-3 Relation Node

An arithmetic node indicates that there exists a certain computable relationship between the arguments. However, we have many other relations that are not arithmetic. The relations between a chemical material and the molecular weight, and a material and the color are such examples. We represent these relationships as predicate nodes in the S.N.. These predicate nodes are called relation nodes. A relation node indicates that there exists a certain de-

pendency relationship between arguments, and it plays the same role as an arithmetic node does in the reasoning process. Though in the case of arithmetic node we know how to compute the value of the unknown argument from the known values of the other arguments, we usually do not know about a relation node the way how to calculate the value of a certain argument from the values of the other arguments. A simple fact like

'All human beings have fathers'

can be expressed by the formula

$$(\forall x)(\exists y)[HUMAN(x) \longrightarrow FATHER(x, y)]$$

This formula is transformed by using a Skolem function into

$$(\forall x)[HUMAN(x) \longrightarrow FATHER(x, f(x))]$$

The Skolem function 'f' in the above formula indicates the dependency relationship of the variable y on the variable x in the first formula. Instead of using Skolem functions, we express such dependency relationship by a relation node. The above formula is expressed in the S.N. as shown in Fig. V-7.

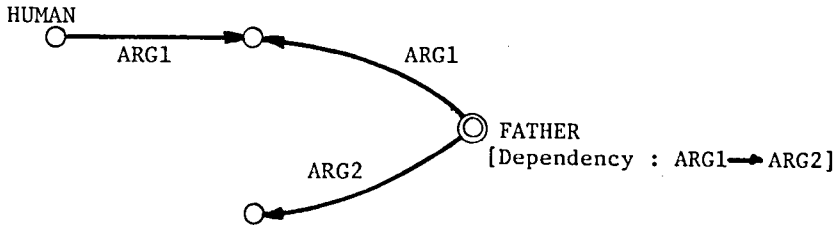


Fig. V-7 Representation of the Fact
$(\forall x)(HUMAN(x) \rightarrow FATHER(x, f(x)))$

In the reasoning process, we need to know not only that a certain dependency relationship exists but also the real correspondence between elements. The real correspondences of elements

are considered to give partial interpretations of the Skolem function. Suppose that we have the following four formulas.

$(\forall x)(\exists y)[HUMAN(x) \longrightarrow FATHER(x, y)]$
FATHER(JIM, JOHN)
FATHER(MIKE, JOHN)
FATHER(TOM, STEVE)

In this case, the first formula is transformed into the Skolem standard formula

$(\forall x)[HUMAN(x) \longrightarrow FATHER(x, f(x))]$

and in the S.N. it is expressed by the network shown in Fig. V-7. The node 'FATHER' is a relation node. The remaining three formulas give the real correspondences between elements. In order to answer a certain type of questions, we can determine who is the father of a given person by using this partial interpretation of the Skolem function. We will assume such an interpretation is stored in an external data base, because the number of these correspondences may be very large in a real application. Therefore, a relation node gives us an entrance from the S.N. to the external data base as an arithmetic node gives us an entrance to an arithmetic algorithm. We assume that external data bases are contructed based on the relational data model. The above four formulas are represented in the S.N. as shown in Fig. V-8.

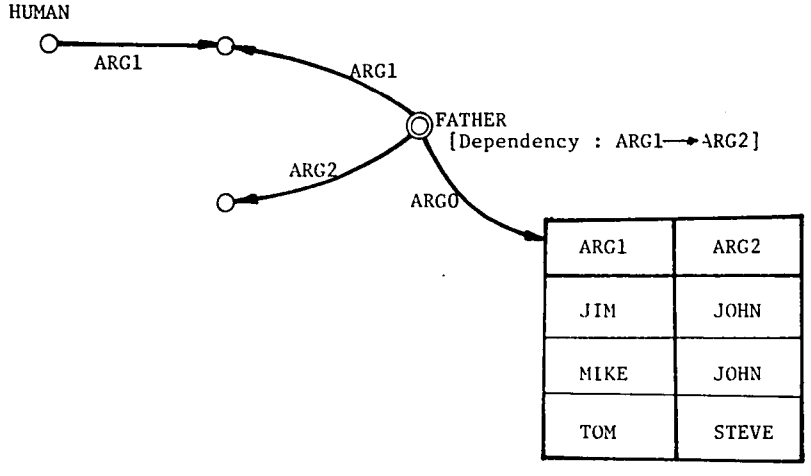


Fig. V-8 Link to an External Data Base

The links attached to a relation node are named ARGi (i=0, 1, 2 ...). The link ARG0 is the special link which points to the name of the table in the external data base. A relation node contains a dependency relationship among the arguments as the internal description of the node.

It is possible to create several relation nodes for a single table. In this case, each relation node describes a different dependency relationship among the columns of the table. Note that the word 'column' has the same meaning as the technical term 'domain' in the relational data base theory. An example in which two relation nodes are created for a single table is shown in Fig. V-9.

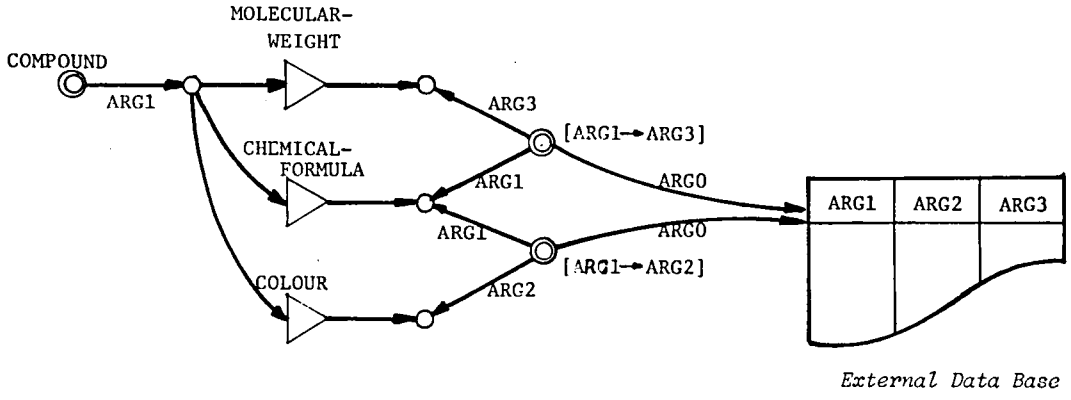


Fig. V-9 Example of Two Relation Nodes for a Single External Data Base

V-3-3-4 Boolean Node

In the reasoning process, the dependency relationship shown by an arithmetic node or a relation node is utilized to determine the unknown values of a certain argument from the known values of the other arguments. On the other hand, a boolean node is mainly evaluated when the values of the all arguments are determined. The result

of evaluation of a boolean node is true or false.

An example of a boolean node is given in Fig. V-10.

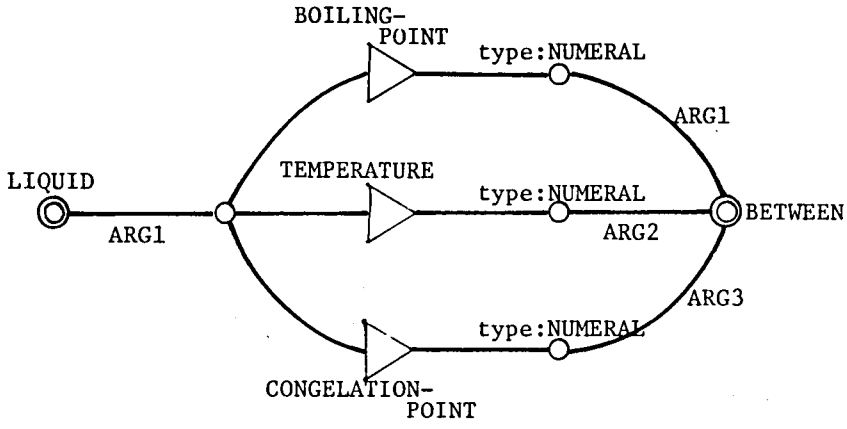


Fig. V-10 Example of a Boolean Node

V-4 Generalization Hierarchy

Current research on semantic networks tends to introduce rigorous notations in networks in order to extend their expressive power. However, introduction of rigorous notations in networks carries with it a cost in the directness of the associative paths for retrieving relevant knowledge. The only advantage of the current semantic network-based approaches over other representational frameworks seems to be that it can retrieve relevant knowledge by utilizing certain hierarchical structures among concepts. We mean here by the word 'concept' a set of objects which satisfy a certain condition. In other representational systems, the only things which can be expressed by such a hierarchical structure are very simple inference rules such as the following.

$$(\forall x)(\forall A)(\forall B)[x \in A \wedge (A \subset B) \longrightarrow x \in B]$$

Note that the relationships $\in$ and $\subset$ are called by different names in different networks. For example, the relationship $\in$ is called by the name such as IS-A, ELEMENT-OF, MEMBER and so on, and the relationship $\subset$ by the name such as SUBSET-OF, SUBCONCEPT, and so on.

Many facts which express certain relationships between concepts but cannot be represented by this simple-minded framework are easily found. In the chemical field we have, for example, the following statements that express certain relationships among concepts.

1. All chemical materials are either gas, liquid or solid.
2. The material whose temperature is between the boiling point and the congelation point is liquid.
3. Hydrochloric acid is a strong acid.
4. The acid which melts copper is either hydrochloric acid, nitric acid or sulphuric acid.

All the above statements express relationships among concepts. But because a simple-minded concept hierarchy cannot express these statements in its framework, one cannot help but express them in ordinary logical formulas. In the S.N., we can embed these facts in the concept hierarchy more naturally and therefore they can be utilized efficiently in the reasoning process. For this prupose, we will introduce the two new types of nodes called SELF and DISJOINT.

V-4-1 Concept Hierarchy and SELF Node

Before we describe the role of a SELF node, we will first mention our concept hierarchy. The following links are used to represent the inclusion relationship among the sets (We follow the notation developed by J.Sowa (1976)).

1. SORT
2. SUBSORT

These two links represent the inclusion relationship between two sets which are specified by two concept predicates. An illustration is given in Fig. V-11.

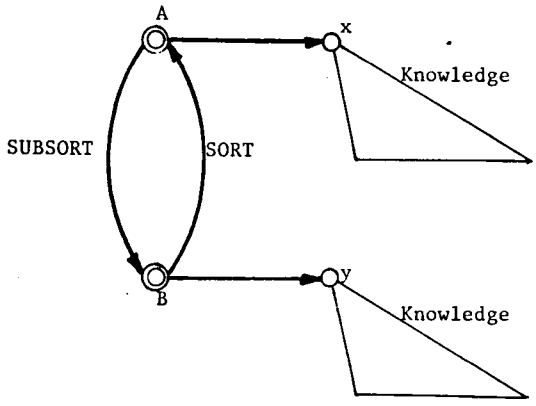


Fig. V-11 Generalization Hierarchy

If a chemical material can be identified with the variable node y in the figure, that is, if a chemical material satisfies the predicate B, then the material can be automatically identified with the concept node A by traversing the SORT link. If a certain thing is identified with a certain variable node, it means that the knowledge attached to the node becomes applicable. A SORT link can be traversed without any checking so that any knowledge attached to the upper concept nodes (the nodes which are linked by SORT links with the current node) is always applicable. However we cannot traverse a SUBSORT link unconditionally.

The concept nodes usually correspond to the sets that are specified by single predicates. However, in some cases the specification of a set is expressed by a noun phrase, that is, a compound formula. The phrase in sentence 2

'material whose temperature is between the boiling point and the congelation point'

and the phrase in sentence 3

'strong acid'

cannot be expressed by single concept predicates, but they indicate useful clusterings of materials. If we can express such clusterings by single concept nodes in the S.N., we can attach any predications to the nodes. Otherwise, we must make predications for each material in the clusterings.

We will extend the definition of a concept node from the restriction that a concept node represent a set specified by a single concept predicate to the definition that a concept node represents a useful clustering of objects, even when the exact members of this cluster may not be known. According to this definition, we can express sentence 2 and sentence 3 can be expressed such as Fig. V-12.

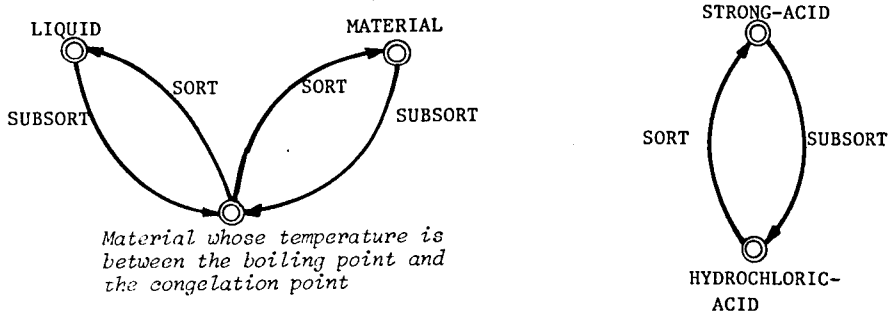


Fig. V-12 Graphical Notations for Sentence 2 and Sentence 3

In this figure, one can see that the concepts which are expressed by the phrases correspond to the single concept nodes in the S.N.. The graphical notations given in Fig. V-12 are, of course, insufficient. We should specify the condition that describes when a certain object can be identified with the concepts.

The TORUS system which has been developed at the University of Toronto utilizes a semantic network for the interface between data bases and casual users. In the semantic network of TORUS, it is possible to attach to each node the function which is called the 'recognition function' for the node. However, we want to express

various kinds of information in the network structure itself as much as possible, because representing knowledge in networks has the following obvious advantage :

A single network can be utilized for various purposes so that the condition for identification can also be used for other purposes. On the other hand, as pointed out by Winograd (1975), in procedural representations a single fact would have to be represented differently for each different purpose.

We use a SELF node for describing the identification condition in the S.N.. Fig. V-13 shows the notations in the S.N. for sentence 2 and sentence 3.

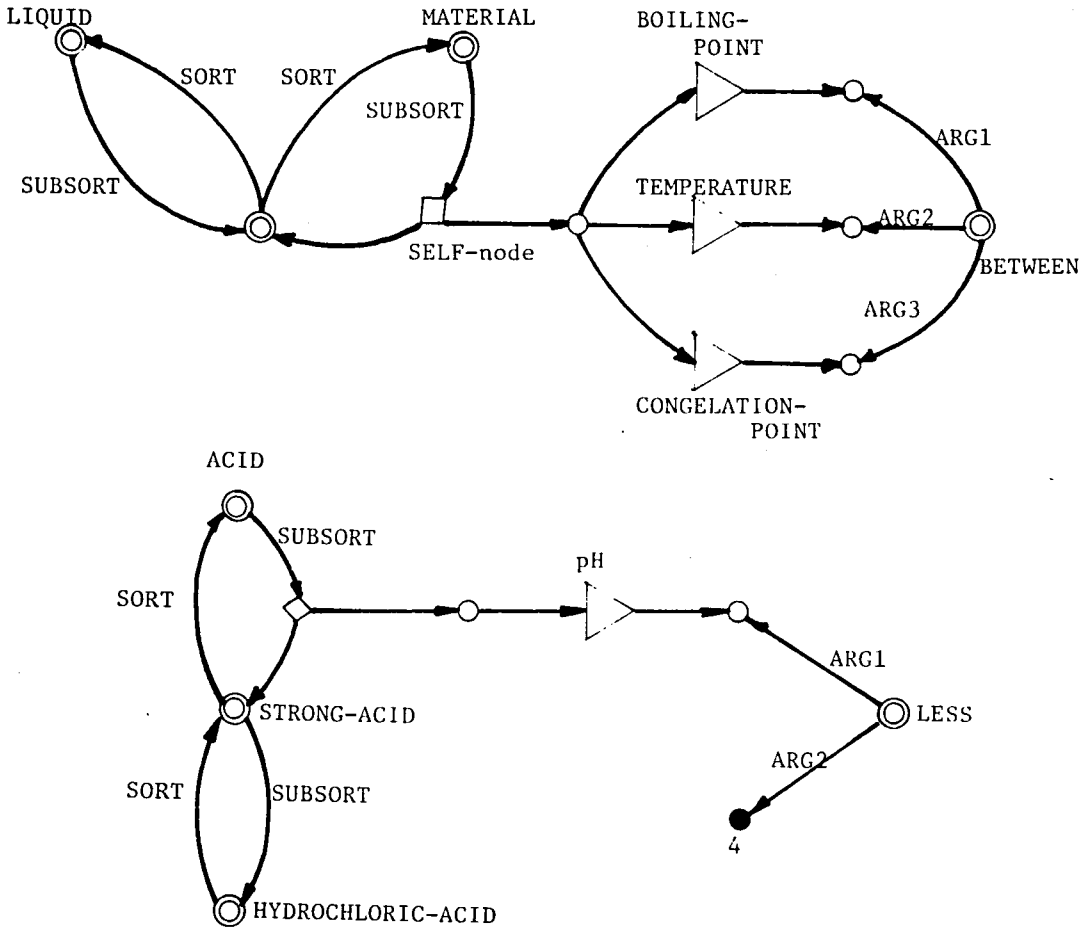


Fig. V-13 The Notations for Sentence 2 and Sentence 3

The format of the description attached to a SELF node is the same as those attached to a concept node. Therefore, when a material is identified with a concept node, the description attached to the SELF node can be utilized in the same manner that the description attached to the concept node is used. For example, when the following two different definitions of concept 'liquid' are given :

'A liquid is a material whose temperature is between the boiling point and the congelation point'

and

'Liquid is a material whose state is *LIQUID'

Then the network becomes like that shown in Fig. V-14. When a material satisfies the description of a SELF node, the description attached to the other SELF node becomes applicable to the material. That is, the network in Fig. V-14 can be interpreted in the following ways:

'If the temperature of a material is between the boiling point and the congelation point, then the material is liquid and the state of the material is *LIQUID.'

'If the state of a material is *LIQUID, then the material is liquid and the temperature is between the boiling point and the congelation point.'

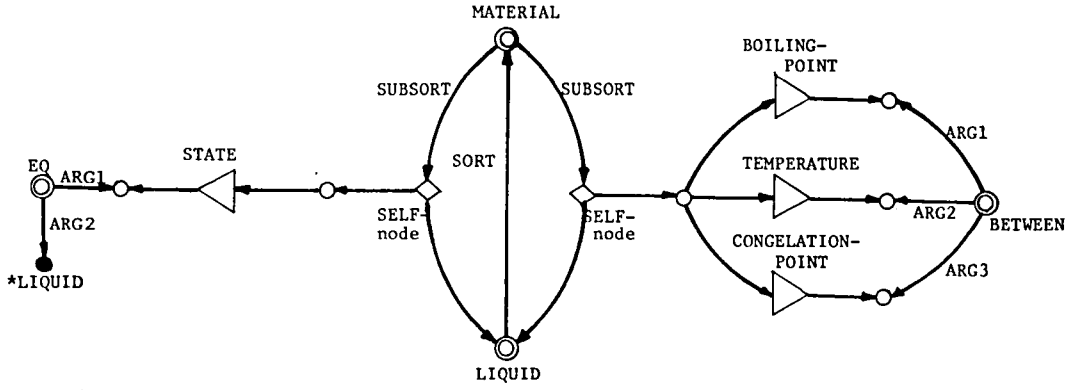


Fig. V-14 Two Different Definitions for the Concept 'Liquid'

V-4-2 Concept Hierarchy and DISJOINT Node

Both sentence 1 and sentence 4 express mutually exclusive relationships among concepts. Suppose we express 'x is material', 'x is liquid', 'x is gas', and 'x is solid' by MATERIAL(x), LIQUID (x) , GAS(x) and SOLID(x), respectively. Then we can express sentence 1 by the formula

$$(\forall x)[\mathrm{MATERIAL}(x) \longrightarrow \mathrm{LIQUID}(x) \vee \mathrm{GAS}(x) \vee \mathrm{SOLID}(x)]$$
$$(\forall x)[\mathrm{LIQUID}(x) \longrightarrow \sim\mathrm{GAS}(x) \vee \sim\mathrm{SOLID}(x)]$$
$$(\forall x)[\mathrm{GAS}(x) \longrightarrow \sim\mathrm{SOLID}(x) \vee \sim\mathrm{LIQUID}(x)]$$
$$(\forall x)[\mathrm{SOLID}(x) \longrightarrow \sim\mathrm{LIQUID}(x) \vee \sim\mathrm{GAS}(x)].$$

And we can also express sentence 4 by the formula

$$(\forall x)(\forall y)[\mathrm{COPPER}(x) \wedge \mathrm{MELT}(x, y) \wedge \mathrm{ACID}(y) \rightarrow \mathrm{HY\text{-}ACID}(x) \vee \mathrm{NI\text{-}ACID} \vee \mathrm{SU\text{-}ACID}(x)]$$
$$(\forall x)[\mathrm{HY\text{-}ACID}(x) \rightarrow \sim\mathrm{NI\text{-}ACID}(x) \vee \sim\mathrm{SU}(x)]$$
$$(\forall x)[\mathrm{NI\text{-}ACID}(x) \rightarrow \sim\mathrm{SU\text{-}ACID}(x) \vee \sim\mathrm{NI\text{-}ACID}(x)]$$
$$(\forall x)[\mathrm{SU\text{-}ACID}(x) \rightarrow \sim\mathrm{NI\text{-}ACID}(x) \vee \sim\mathrm{HY\text{-}ACID}(x)].$$

The predicates used here have obvious meanings.

Though there often appear the mutually exclusive relationships such as those expressed in sentence 1 and sentence 4, the logical formulas for expressing them are very complex, and therefore it may be very time consuming for the general proof procedure to utilize them during the deduction.

We call a set of concepts 'disjoint set', if the concepts in the set satisfy the following conditions :

1. If an object is identified with a certain concept in the set, the object can never be identified with the other concepts in the set
2. If an object cannot be identified with the other concepts in the set except one, the object is automatically identified with it.

In the S.N., we represent a disjoint set by a node called DISJOINT node. Because the set (liquid, gas, solid) in sentence 1 and the set (hydrochloric acid , nitric acid, sulphuric acid) are

disjoint sets, sentence 1 and sentence 4 are represented in the S.N. as shown in Fig. V-15. DISJOINT node are often useful to inhibit assertions and general axioms that are semantically irrelevant to the search from entering into the deductive search(see section V-8-1).

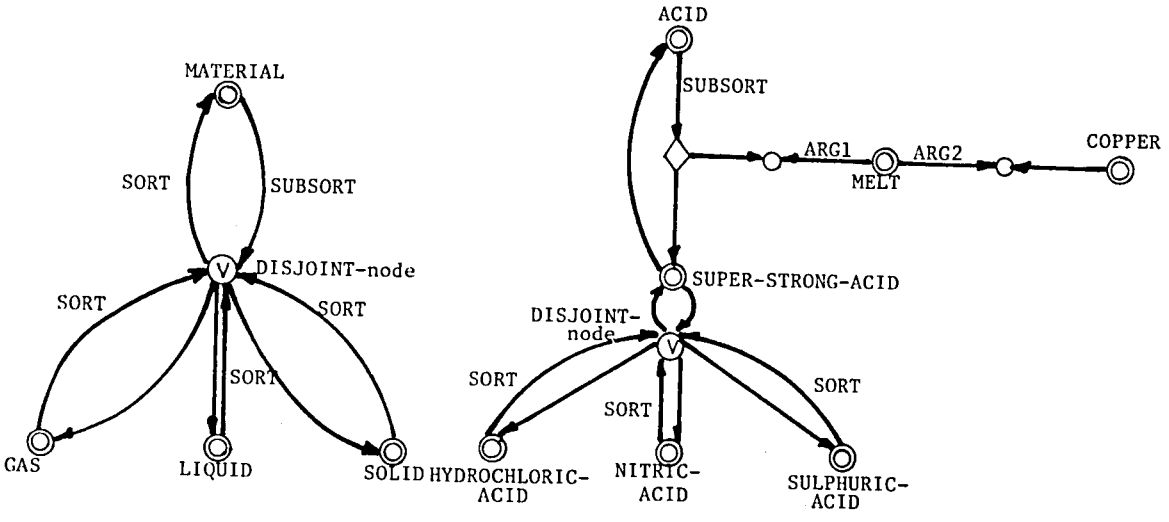


Fig. V-15 Examples of DISJOINT - nodes ——— Representation of Sentence 1 and Sentence 4

V-5 Path Merging and Assimilation of New Knowledge

The procedure for problem solving or deduction is to traverse the network to find knowledge relevant to the current problem. If a certain object in the current problem is identified with a certain concept node in the S.N., the knowledge attached to the node becomes applicable to the object. However, there are many paths (sequence of nodes and links) leaving from a concept node. The more independent paths leave from a node, the less searching is efficient. A new statement should be properly assimilated with the statements already stored in the network. The process called 'path merging' is per-

formed, when a new piece of knowledge is entered.

In the S.N., the predicate nodes except for concept nodes express certain relationships among their arguments. The path from a concept node to one of these nodes gives a path of reference to the relationship. Therefore, the process of path merging is applied to the sequence of concept nodes, variable nodes and function nodes. A simple example is shown in the following. Suppose that the following knowledge has already been stored in the S.N..

<mass of a solution> = <mass of the solute> + <mass of the solvent>

And the new knowledge

<density of a solution>=<mass of the solute> /<mass of the solution>

is entered.

The stored knowledge is in the form shown in Fig. V-16(a). The new knowledge can be represented by the network given in Fig. V-16(b).One can recognize that the two paths in the new knowledge are identical to those already stored in the S.N.. That is, the node v' and w' in Fig. V-16(b) correspond to the following formulas.

$$\left.\begin{array}{l} v' = \text{mass}\ (x) \\ w' = \text{mass}\ (\text{solute}(x)) \end{array}\right| \text{SOLUTION}(x)$$

On the other hand, the nodes v and w also represent the concepts such as (in Fig. 16(a))

$$\left.\begin{array}{l} v = \text{mass}\ (x) \\ w = \text{mass}\ (\text{solute}(x)) \end{array}\right| \text{SOLUTION}(x)$$

By considering these formulas, it is obvious that v and v', w and w' are identical. Therefore, the paths can be merged. The resultant structure is shown in Fig. V-17.

Thus the two pieces of knowledge which are given separately are combined together in the resultant network, and the various relationships among the concepts <mass of a solution>, <density of a solution>, <mass of the solvent> and <mass of the solute> are appropriately represented so that one can get the desirable formula

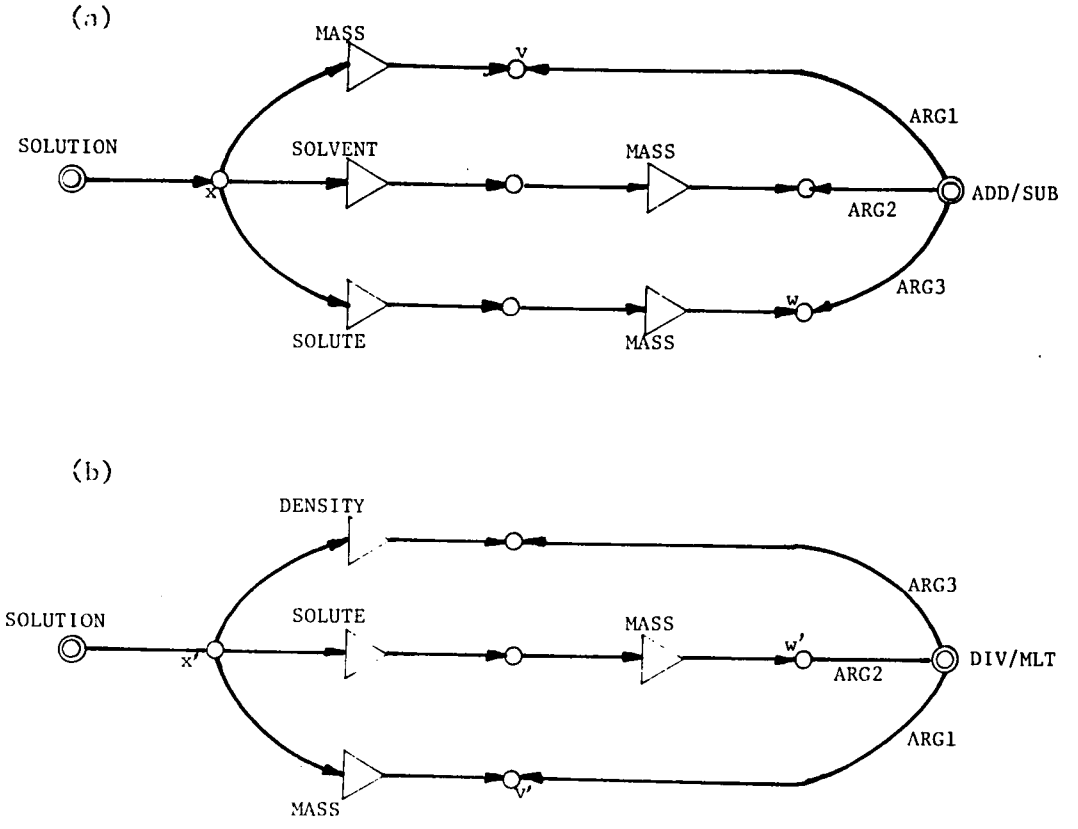


Fig. V-16 Stored Knowledge and New Knowledge

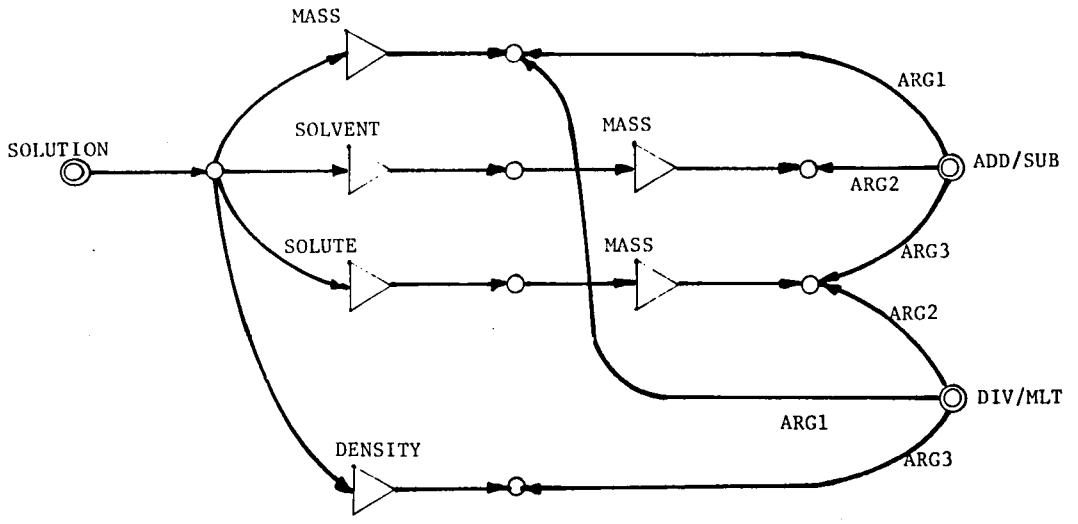


Fig. V-17 Assimilated Form

from this network without any transformation of formulas. For example, if one wants to calculate the density of a solution from the mass of the solvent and the solute , the following formula is easily generated from the network.

<density of a salution>=<mass of the solute> /(<mass of the solvent>+<mass of the solute>)

V-6 Generic Node and Occurrence Node

There is an important distinction of nodes in the S.N.. That is the distinction between the generic nodes and the occurrence nodes. The generic node for a predicate contains general information or a definition of the predicate i.e., how to evaluate the expressions that contain the predicate, when the predicate is satisfied, what knowledge becomes applicable if the predicate is satisfied, and so on. On the other hand, an occurrence node for a predicate is the node which indicates that the predicate is used in a certain statement. It is easy to understand the distinction between a generic node and an occurrence node for a concept. Because a concept node in the generization hierarchy gives the definition of a certain concept, it is a generic node for the concept. On the contrary, the concepts 'liquid' and 'water' in the statement such as 'solvent of a solution is liquid', 'if the solvent of a solution is water, the solution is aqueous solution' play the same roles as boolean predicates. These statements do not give the definition of liquid and water. So the concept liquid in these statements is expressed in the form of occurrence nodes.

Representation of the statement 'The solvent of a solution is liquid' is given in Fig. V-18. The node x in this figure is an occrrence node which shows the predicate LIQUID is used there. Without the occurrence node x, the network looks like Fig. V-19. In this

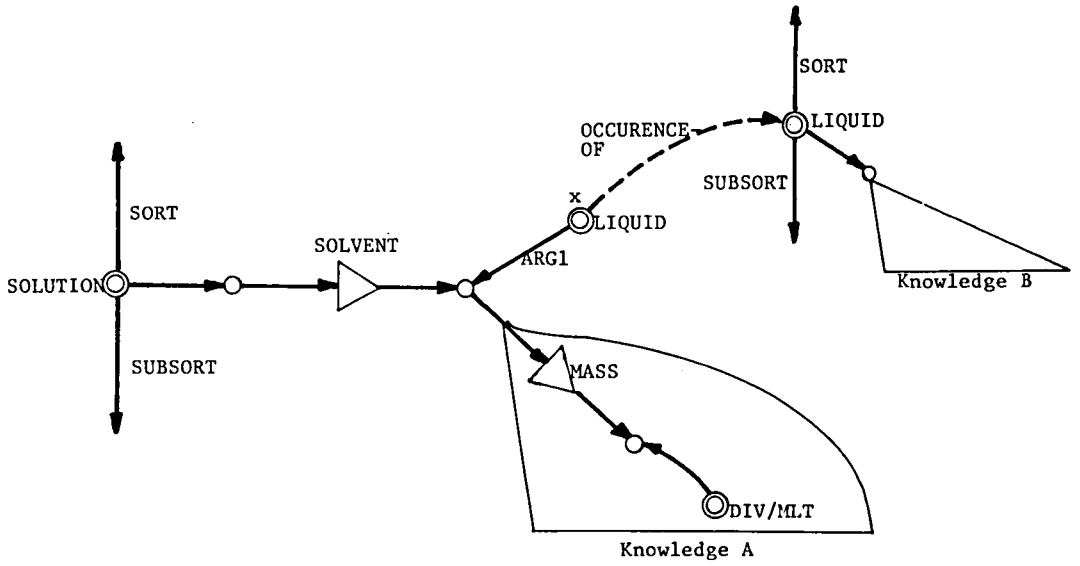


Fig. V-18 Distinction between Generic Nodes and Occurrence Nodes

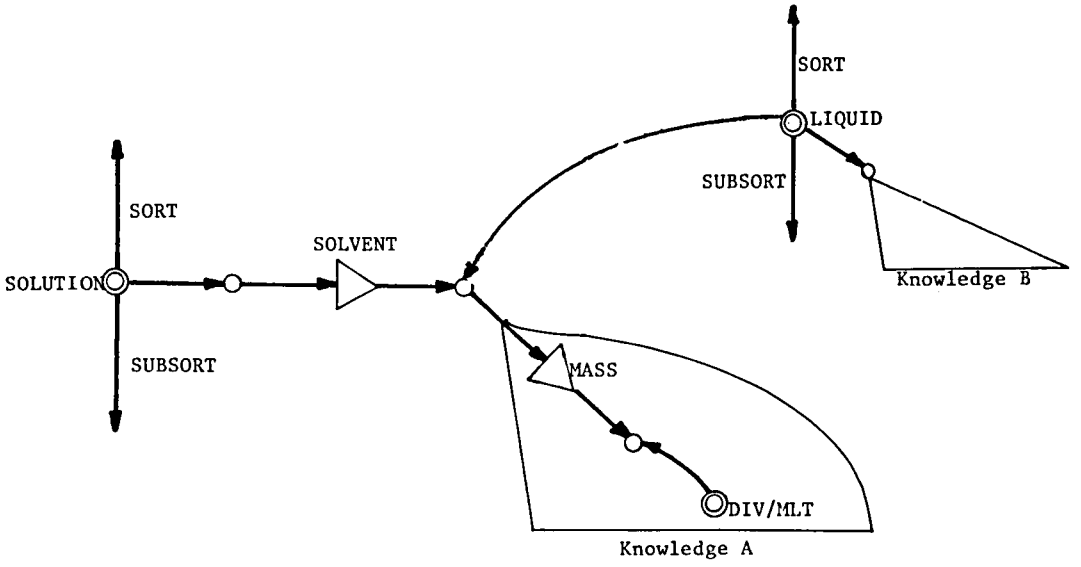


Fig. V-19 Network without Occurrence Nodes

network we cannot distinguish the applicability conditions of knowledge A and knowledge B. Knowledge A consists of the predications which are applicable only to the object which is the solvent of a certain solution, while knowledge B consists of the predications which are applicable to liquid in general. As shown in Fig. V-18, an occurrence node is linked to the generic node with an occurrence-of link. By traversing this link, the problem solver utilizes any knowledge which is attached to the generic node. Therefor both knowledge A and knowledge B are applicable to the object which is the solvent of a solution. On the contrary, because it is prohibited to traverse an occurrence-of link from the generic node to a certain occurrence node, knowledge A is not applied to liquid in general.

Fig. V-20 shows another example of occurrence nodes.

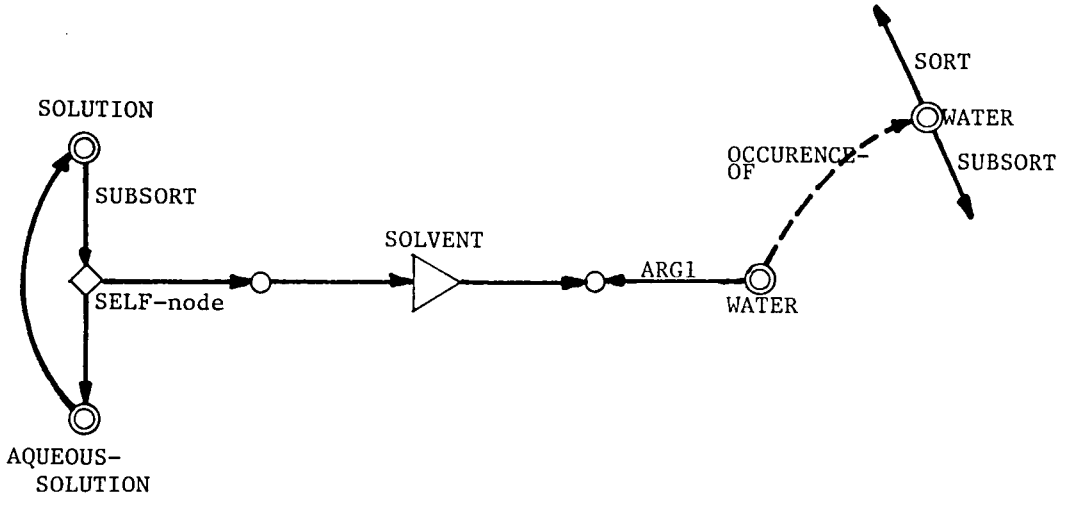


Fig. V-20 Example of an Occurrence Node in a SELF Description

In this example an occurrence node appears in a SELF description. Suppose that a certain object has already been identified with SOLUTION. And suppose that from some reasons the problem solver is going to identify it with AQUEOUS-SOLUTION. In this case, the occurrence node plays the role of a boolean predicate. That is, the problem solver tries to prove that the solvent of the solution is water. Not only the knowledge which are applicable to water in general but

also the methods how to prove that an object is water are attached to the generic node. Those methods will be tried.

The boolean nodes such as BETWEEN, GREATER et. al. and the arithmetic nodes such as ADD/SUB, DIV/MLT et. al. which appear in the networks given previously are all occurrence nodes. The generic node for an arithmetic node contains the procedure which calculates the unknown value from known ones, while the generic node for a boolean node contains information about how to evaluate the predicate. The relation nodes in the network given so far are all generic nodes, because they contain the information about the corresponding external data bases. That is, they describe which external data bases are relevant, how to access the data bases and so on.

V-7 Description Language for the S.N.

For representing all the knowledge in a certain specific domain such as the field of chemistry, the size of the S.N. becomes so large that one cannot represent it all at once. It would be convenient if the user could express his knowledge as he normally expresses it in natural languages, and the system were able to construct the network with data gathered from inputs separately given by the user. We provide a convenient programming language for defining networks. Though the language by no means looks like natural languages such as Japanese, it has various characteristics in common with natural languages. The user may express his knowledge in highly descriptive forms. In this section, we will give the specifications of the language.

Each concept node in fact represents a class of objects. And therefore, each concept node must contain the criteria to decide whether a certain object belongs to the class or not. In section V-4, we use the word 'generalization hierarchy' to describe the relation-

ship between concepts. During the deduction process, the generalization hierarchy plays an important role in selecting relevant knowledge. It also plays an important role in defining a concept. One often describes a concept in comparison with the other concepts, especially its super-concepts. In the description language, one can define a concept by specifying what conditon is necessarily satisfied for an object to be identified with the concept. The specification of the condition is often done by comparing the concept with the super-concepts. The followings are simple examples of definition of concept nodes.

Example 1

```
(CONCEPT        NAME= LIQUID
    SORT=  [(CONCEPT  NAME= MATERIAL)
                (BETWEEN    (!  BOILING-POINT)
                            (!  TEMPERATURE)
                            (!  CONGELATION-POINT))]   )
```

Example 2

```
(CONCEPT        NAME= AQUEOUS-SOLUTION
    SORT=  [(CONCEPT  NAME= SOLUTION)
                ( (CONCEPT  NAME= WATER) (!  SOLVENT))]  )
```

The above two examples are represented in the S.N. as shown in Fig. V-14 and Fig. V-20 respectively. The super concepts of LIQUID and AQUEOUS-SOLUTION in the above examples are MATERIAL and SOLUTION, respectively. The framed expressions in the above definitions are the expressions which specify the conditions when an object that has already been identified with the super-concept can be identified with the concept. Each concept can have a unique name for the later reference to the node. The expression NAME= gives the unique name to the concept. Note that a concept node can be used as a boolean predicate. The expression (CONCEPT NAME= WATER) in the second example plays the same role as the boolean predicate BETWEEN does in the first example. The arguments of BETWEEN and (CONCEPT NAME= WATER) are designated by the paths. A "!" in the path descriptions refers to the current concept,

and BOILING-POINT, TEMPERATURE, CONGELATION-POINT, and SOLVENT are all the names of functions.

We can define a concept without a unique name.

Example 3

```
(CONCEPT
   SORT=  [(CONCEPT  NAME= ACID)
                    (LESS (! pH)  5)   ] )
```

In this example, the acid whose value of pH is less than 5 is defined as a sub-concept of ACID. Any knowledge can be attached to this concept, if it is applicable to the object which is acid and whose pH value is less than 5. Because we can use the expression (CONCEPT ....) recursively in the defintion of concepts, the following expressions are admissible.

Example 4

```
(CONCEPT      NAME= NITRIC-ACID
    SORT=  [(CONCEPT     NAME= STRONG-ACID
                 SORT=  [(CONCEPT     NAME=ACID)
                              (LESS (!  pH)  3)]
                  (  .....................  )]
```

In the earlier example, the predicates in the condition is rather simple. However, one can also specify the condition by using arbitrary concept descriptions instead of (CONCEPT NAME=WATER).

As mentioned earlier, some concepts have more than one super-concept. Several SORT descriptions can be given for a single concept such as shown in the following.

```
SORT= [(Super-concept)  (SELF-description)]
      [(Super-concept)  (SELF-description)]
      ..................................
      [(Super-concept)  (SELF-description)]
```

Disjoint sets are also declared in the definition of a concept.

The following is an example.

Example 5

```
(CONCEPT      NAME= MATERIAL
    SUBSORT=  [(CONCEPT NAME=SOLID)(EQ (! STATE) *SOLID)]
              [(CONCEPT NAME=LIQUID)(EQ (! STATE) *LIQUID)]
              [(CONCEPT NAME=GAS)(EQ (! STATE) *GAS)]
    DISJOINT= (SOLID, LIQUID, GAS)            )
```

Some dependency relationships between various properties of a concept can be given in the definition of the concept node by using the path descriptions.

Example 6

```
(CONCEPT      NAME= SOLUTION
    COMPUTATION ;
        (! MASS)=(! SOLUTE MASS) + (! SOLVENT MASS)
        (! DENSITY)=(! SOLUTE MASS) / (! MASS)
```

Example 7

```
(CONCEPT      NAME= COMPOUND
    RELATION ;
        (! MOLECULAR-WEIGHT)= [TABLE= TAB]
                              (/    (!  CHEMICAL-NAME)   *)  )
```

The S.N. representations of the above examples are shown in Fig. V-17 and Fig. V-9 respectively.

There exists a certain kind of useful knowledge which describes only incidental relationships and may not be directly useful for calculating or searching for unknown values from known ones.

Example 8

```
(CONCEPT      NAME= ALCOHOL
    DECLARATION ;
        (LESS  (! BOILING-POINT) ((CONCEPT NAME= WATER)
                                  BOILING-POINT))  )
```

The second argument of GREATER in the above is an example of the path which starts from a concept other than the current concept. The network for this example is given in Fig. V-21.

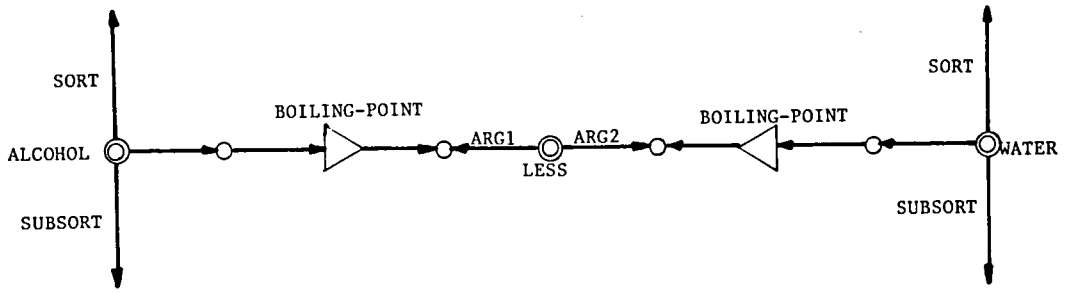


Fig. V-21 Example of Representations of Incidental Relationships

V-8 Operations of the S.N.

The problem solver performs its task by traversing the S.N.. There are several basic modes of reasoning in the S.N.. In this section, we will describe these basic operations. The validity of each operation will be examined by considering the corresponding logical formulas.

V-8-1 Identification of Objects with Concepts

During the problem solving process, the system is often required to identify an object with a certain concept. Identification of objects with the concepts in the S.N. will be performed in the following cases.

1. When a problem is given to the system, the system first begins to locate in the S.N. the objects which appear in the problem description. This is carried out in order to restrict the applicable knowledge to these objects which are known to be pertinent.

2. Concept descriptions are sometimes used as boolean predicates in the S.N.. In this case, identification is carried out for proving that a certain object satisfies the identification condition of some concept.

In both cases, identification of an object with a concept in the S.N. requires proving that an object satisfies the predicate which corresponds to the concept. We will explain here the operations by considering the following simple example.

We have the description

```
(CONCEPT   NAME= SOLUTION
   ..............................
   SUBSORT : ((CONCEPT   NAME= AQUEOUS-SOLUTION
                         ((CONCEPT   NAME= WATER) (! SOLVENT)))
   ..............................  ).
```

The problem description is

```
(OBJECT-1     NAME= SOLUTION
    ........................
    (!  SOLVENT)= (OBJECT-2   NAME= WATER)
    ........................)
```

Because of the existence of the expression NAME= SOLUTION in the description of the object O_1 , O_1 can be identified with the concept SOLUTION in the S.N.. We will next try to prove whether O_1 can be identified with one of the sub-concepts of SOLUTION. In the description of the concept SOLUTION, we find AQUEOUS-SOLUTION as a subconcept of SOLUTION, and also find the additional condition for traversing from SOLUTION to AQUEOUS-SOLUTION. This condition is

((CONCEPT NAME = WATER)(! SOLVENT)), that is, if the solvent of a solution can be identified with the concept WATER, the solution is acceptedas an AQUEOUS-SOLUTION. Because ! in the path description represents the current object that is identified with the concept, we will try to prove ((CONCEPT NAME = WATER)(O1 SOLVENT)) is true. The result of the evaluation (O1 SOLVENT) becomes O2. In the above example, it is easy to prove that O2 is WATER because the object O2 can be identified immediately with the concept WATER by using the description (OBJECT-2 NAME = WATER) in the problem description. As the result, the solution O1 can be identified with AQUEOUS-SOLUTION.

In the general situation in which the description of the object O2 is (OBJECT-2 NAME = C0 .....). The following operations will be performed in order to identify the object O2 with the concept WATER.

STEP 1. The object O2 is located in the S.N. by using the NAME = description. That is, the object will be identified with the concept C0 in the S.N..

STEP 2. The super concepts of the object O2 are listed up by traversing SORT-links from C0. The list of the super concepts is noted as (C1, C2, C3, ..., Cn). (See Fig. V-22).

STEP 3. If the concept WATER is in the list, ((CONCEPT NAME = WATER) O2) is true (see Fig V-23). Otherwise, go to the next step.

STEP 4. The super concepts of the concept WATER are enumerated in turn by traversing SORT-links from WATER. If a concept Ci in (C1, C2, ..., Cn) is encountered, then Ci is a common super concept of the object O2 and the concept WATER. Go to Step 5. If no such common super concepts can be found, the predicate ((CONCEPT NAME = WATER) O2) is assigned NIL.

STEP 5. The conditions attached to the SUBSORT-links from the common super concept Ci to the concept WATER will be evaluated in turn. If all the conditions are satisfied by the object O2, then O2 can be considered as an instance of the concept WATER (see Fig. V-24).

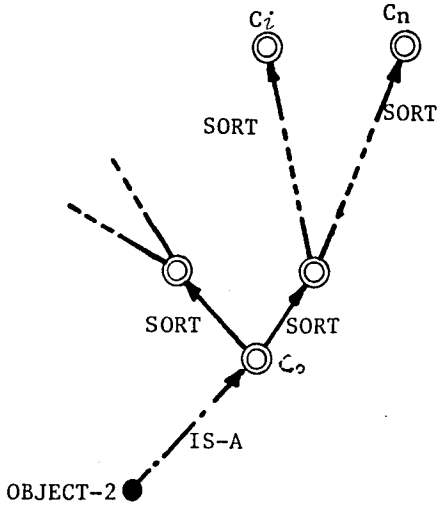


Fig. V-22 Step 2

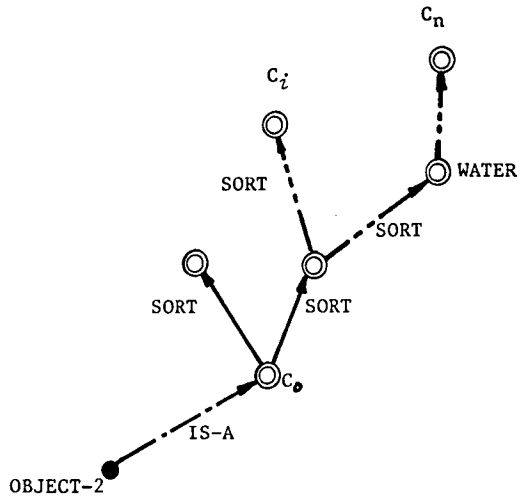


Fig. V-23 Step3(WATER is one of the super concept of C_o)

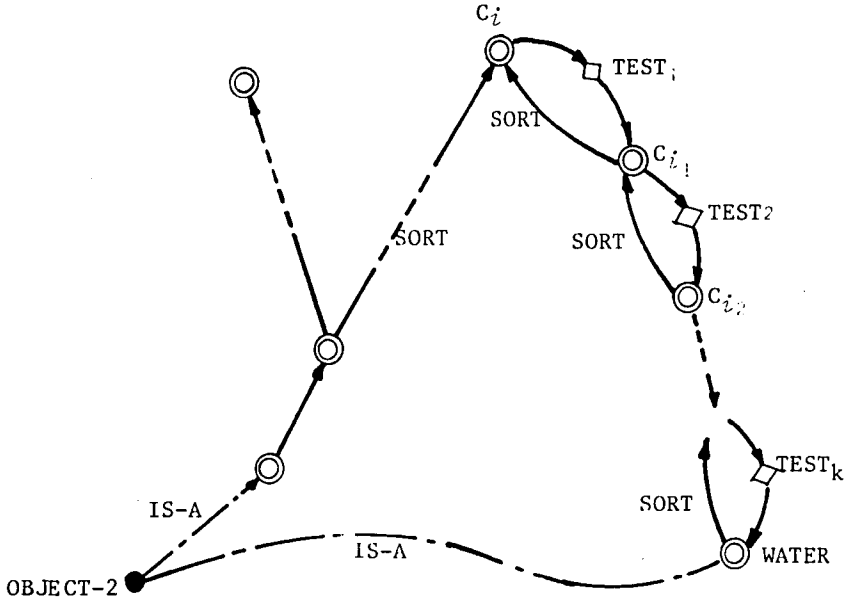


Fig. V-24 Step 5(C_i is the common super concept of O_2 and WATER)

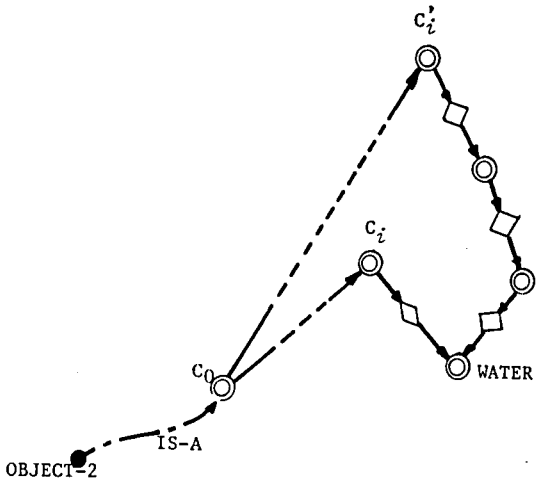


Fig. V-25 The path from C_i to WATER will be tried at Step 5 in the algorithm, because it is natural to expect that the proof path from C_i to WATER is easier to be accomplish than the proof path from C_i' to WATER.

((CONCEPT NAME = WATER) O_2) becomes T. Otherwise it becomes NIL. If there are several such common super concepts, the concept which has the shortest path length from the concept WATER will be tried first. If this fails, then the next shortest path will be tried next and so on. In such a way, the control tries first the proof path which seems the easiest (see Fig. V-25).

If ((CONCEPT NAME = WATER) O_2) becomes T, the object O_1 can be identified with the concept AQUEOUS-SOLUTION. That is, ((CONCEPT NAME = AQUEOUS-SOLUTION SORT = [(CONCEPT NAME = SOLUTION)((CONCEPT NAME = WATER)(! SOLVENT))]) O_1) becomes T. Thus the identification process is performed recursively.

We will explain the validity of the above whole process by examining the corresponding logical operations. Let us begin from the point at which initial problem, to identify O_1 with the concept AQUEOUS-SOLUTION has been already established. At first, we do not know what knowledge in the S.N. is relevant to the current problem. So the initial problem can be formulated by the following formulas.

$$\begin{array}{l} \text{SOLUTION}(O_1) \\ \quad ? \end{array} \quad \Big| \vdash \quad \text{AQUEOUS-SOLUTION}(O_1) \quad ?$$

By examining the SELF-node description attached between the concept SOLUTION and the concept AQUEOUS-SOLUTION, we notice the fact that if the solvent of the solution is water, then the solution is AQUEOUS-SOLUTION. That is,

$$\begin{array}{l} \text{SOLUTION}(O_1) \\ (\forall x)[\text{SOLUTION}(x) \wedge \text{WATER}(\text{solvent}(x)) \rightarrow \text{AQUEOUS-SOLUTION}(x)] \\ \quad ? \end{array} \quad \Big| \vdash \quad \begin{array}{l} \text{AQUEOUS-} \\ \text{SOLUTION } (O_1) \\ ? \end{array}$$

We search through the problem description to find the solvent of O_1. It is O_2. And in the step 1 described above, O_2 is identified with the concept C_0. So the problem is transformed into

$$\begin{array}{l} \text{SOLUTION}(O_1) \\ (\forall x)[\text{SOLUTION}(x) \wedge \text{WATER}(\text{solvent}(x)) \rightarrow \text{AQUEOUS-SOLUTION}(x)] \\ \text{EQ}(\text{solvent}(O_1), O_2) \\ C_0(O_2) \\ \quad ? \end{array} \quad \vdash \quad \text{AQUEOUS-SOLUTION}(O_1) \ ?$$

The reasoning process which is performed in Step 1, 2 and 3 roughly corresponds to the following logical operations if it succeeds at Step 3.

$$\begin{array}{ll} & C_0(O_2) \\ & (\forall x)(C_0(x) \rightarrow C_1(x)) \\ (A) & (\forall x)(C_1(x) \rightarrow C_2(x)) \\ & \quad \vdots \\ & (\forall x)(Ci(x) \rightarrow \text{WATER}(x)) \end{array} \quad \vdash \quad \text{WATER}(O_2)$$

And if the reasoning process succeeds at Step 5, the process in Step 4 and 5 corresponds to the logical operations:

$$\begin{array}{ll} & (C_0)(O_2) \\ & (\forall x)(C_0(x) \rightarrow C_1(x)) \\ & (\forall x)(C_1(x) \rightarrow C_2(x)) \\ & \quad \vdots \\ (B) & (\forall x)(C_{i-1}(x) \rightarrow C_i(x)) \\ & (\forall x)[(C_i(x) \wedge \text{TEST}_1(x) \rightarrow C_{i_1}(x))] \wedge \text{TEST}_1(O_2) \\ & (\forall x)[(C_{i_1}(x) \wedge \text{TEST}_2(x) \rightarrow C_{i_2}(x))] \wedge \text{TEST}_2(O_2) \\ & (\forall x)[(C_{i_{k-1}}(x) \wedge \text{TEST}_k(x) \rightarrow \text{WATER}(x))] \wedge \text{TEST}(O_2) \end{array} \quad \vdash \quad \text{WATER}(O_2)$$

Here TEST_j represents the SELF-node description which is attached to the SUBSORT-link between the concept node $C_{i_{j-1}}$ and C_{i_j} ($C_{i_0} = C_i$,

C_{i_k} = WATER).

If one of the above two operations succeeds and WATER(O_2) is proved, then the initial problem has completely solved. The whole process can be expressed by the following logical formulas.

SOLUTION(O_1),

(∀x)[SOLUTION(x)∧WATER(solvent(x)) ⟶ AQUEOUS-SOLUTION(x)] ⊢

EQ(solvent(O_1), O_2)

(A) or (B)

AQUEOUS-SOLUTION(O_1)

From the above discussions, one will be able to understand how both the processes of selecting relevant knowledge and performing deduction are carried out by traversing the network. It will be also well recognized what role the SELF node descriptions play in the deduction. We have another group of nodes in the generalization hierarchy, i.e. the DISJOINT nodes. In the following, we will describe how the problem solver treats this type of node.

In order to include DISJOINT nodes in the previous algorithm, we will revise the step 2 and step 4 as follows.

STEP 2 The super concepts of object O_2 are listed up by traversing SORT-links from C_0. The list of the super concepts is noted as (C_1 C_2 ..... C_n). And the DISJOINT nodes which are encountered during the traverse are also listed. The list of the DISJOINT nodes is called DISJOINT-list.

STEP 4 The super concepts of the concept WATER are enumerated in turn by traversing SORT-links from WATER. If a concept C_i is encountered which is also super concept of O_2, then C_i is the common super concept of the object O_2 and the concept WATER. Go to Step 5. If a DISJOINT node is encountered which is in the DISJOINT-list created at Step 2 before a common super-concept is encountered, then the proof of ((CONCEPT NAME = WATER) O_2) fails. That is, ((CONCEPT NAME = WATER) O_2) = NIL. If no such common supper concepts can be found,

((CONCEPT NAME = WATER) O_2) becomes NIL.

Because a DISJOINT node represents a certain disjoint set and the concepts which are the members of the disjoint set do not have any objects in common, the meaning of the revised part in Step 4 can be seen from Fig. V-26.

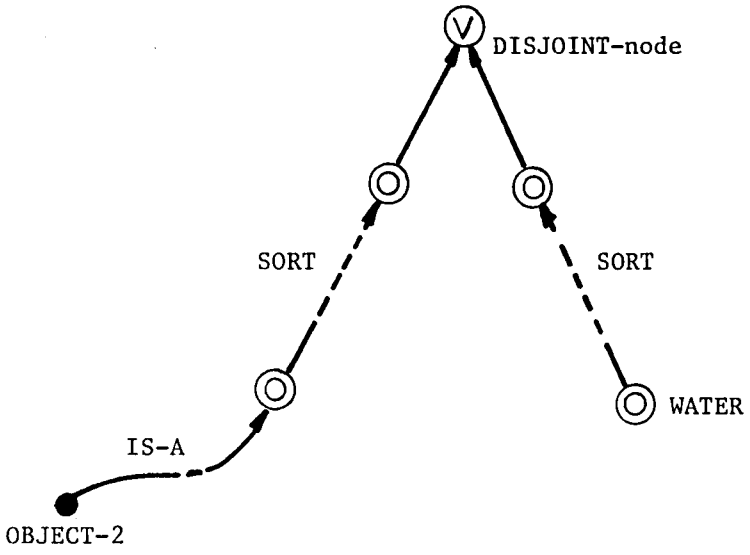


Fig. V-26 The Role of a DISJOINT-node in the Deduction Process

By this revision, we can avoid further futile operations which we may have done otherwise whenever we encounter a common DISJOINT node. The revised portion in Step 4 roughly corresponds to the following mode of reasoning.

$$\frac{C_0(O_2)}{(\forall x)(C_0(x)\rightarrow C_1(x))}$$

$$\cdots\cdots\cdots\cdots\cdots$$

$$\frac{(\forall x)(C_i(x)\rightarrow C_j(x))}{(\forall x)(\mathrm{WATER}(x)\rightarrow C_k(x))}$$

$$\cdots\cdots\cdots\cdots\cdots$$

$$\frac{(\forall x)(C_\ell(x)\rightarrow C_m(x))}{(\forall x)(C_m(x)\rightarrow \sim C_j(x)\wedge\ldots)}$$

$$\vdash \sim \mathrm{WATER}(O_2)$$

V-8-2 Calculation of Property Values

When an object is identified with a certain concept, the knowledge attached to the concept node becomes applicable to the object. By using the knowledge we can calculate the values of certain properties of the object. We will show the operations on the S.N. for the calculation of values by considering the following simple example.

Suppose the following descriptions are given.

```
(CONCEPT     NAME = SOLUTION
      ......................
   COMPUTATION : (!  MOL-DENSITY) = (!  SOLUTE MOL)/(!  VOLUME)
      ...................... )
(CONCEPT     NAME = CHEMICAL-MATERIAL
      ......................
   COMPUTATION : (!  MOL) = (!  MASS)/(!  MOLECULAR-MASS)
      ......................
   RELATION : (!  MOLECULAR-MASS) = [TABLE = TAB]
                                    (/  /  (!  CHEMICAL-NAME)  *  /)
      ......................)
(CONCEPT    NAME = SODIUM
      ......................
   DECLARATION : (EQ (!  CHEMICAL-NAME)  'SODIUM)
      ......................)
```

Suppose we will also be informed by traversing SORT-links that SODIUM is a sub-concept of CHEMICAL-MATERIAL. The corresponding S.N. is shown in Fig. V-27. The description about CHEMICAL-MATERIAL shows that the mol of a certain CHEMICAL-MATERIAL can be calculated from the values of the mass and the molecular-mass of the chemical-material. It also shows that the molecular-mass can be discovered from the external data base named TAB if the chemical-name is known.

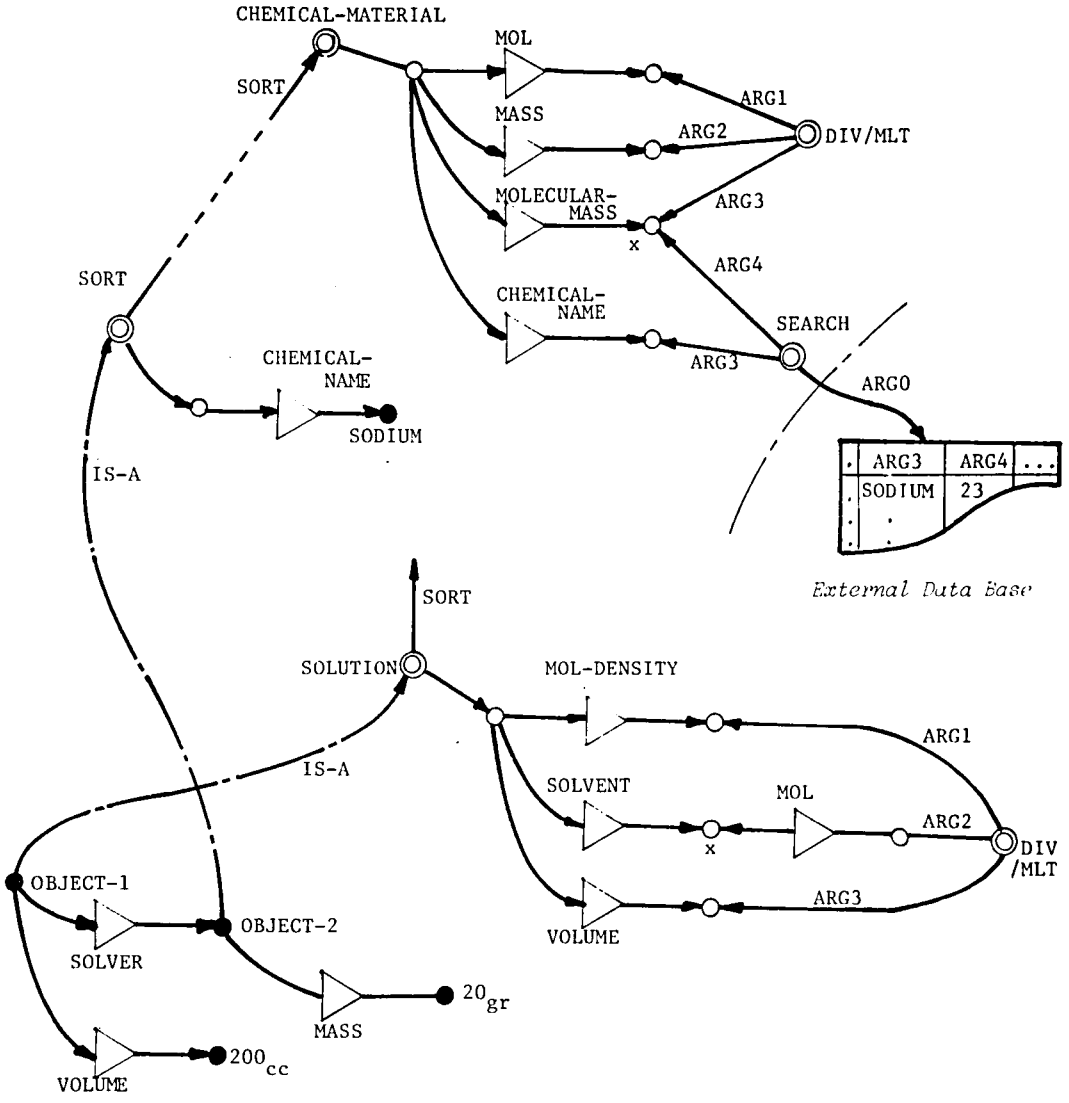


Fig. V-27 Example of Calculation of Property Values

The problem description is given as follows.

```
(OBJECT-1      NAME = SOLUTION
     (!  SOLUTE) = (OBJECT-2      NAME = SODIUM
                          (!  MASS) = 20 gr.        )
     (!  VOLUME) = 200 cm³                          )
```

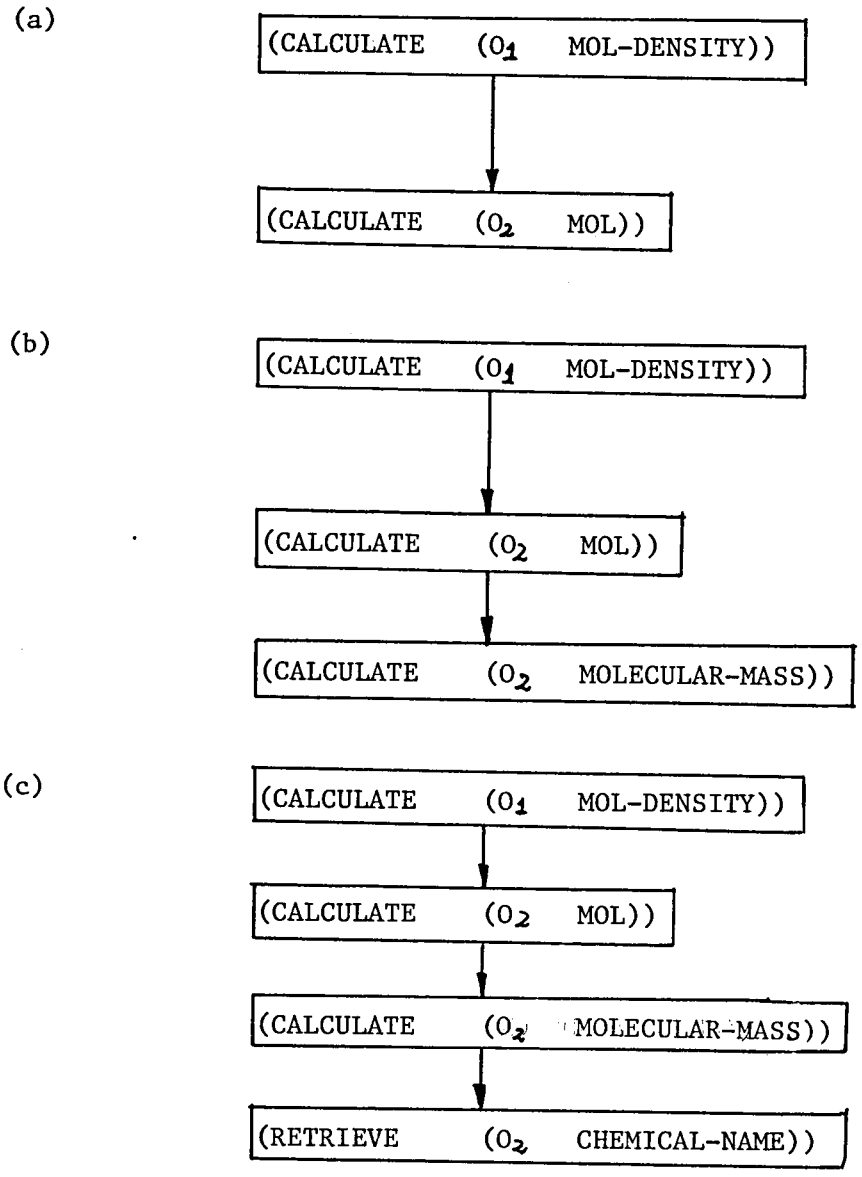


Fig. V-28 Problem-Subproblem Hierarchy

The description tells us that there exists a certain solution, that the solute of the solution is SODIUM, that the mass of the solute is 20 gr and that the volume of the solution is 200 cm^3. Suppose that we want to calculate the value of the mol-density of the solution. By the description NAME = SOLUTION and NAME = SODIUM, the object-1 and object-2 are identified with the concept SOLUTION and SODIUM, respectively. Because a calculation method of mol-density is attached to the concept SOLUTION, the problem-solver tries the method first. In order to calculate the mol-density of a solution, this method requires the values of the mol of the solute of the solution and the volume of the solution. That is, in order to calculate the mol-density of O_1 we need to know both the mol of the solute of O_1 and the volume of O_1. By traversing the network given by the problem description, it is discovered that the volume of O_1 is 200 cm^3 and the solute of O_1 is O_2. However, at this point the mol of the solver, that is, the mol of O_2 is still unknown. The subproblem (CALCULATE (O_2 MOL)) is created for calculating the mol of O_2 (see Fig. V-28(a)).

Because the object O_2 is linked to the concept SODIUM by an IS-A link, the knowledge attached to this concept is applicable. However, we cannot find any knowledge for the calculation of the mol of O_2 in this set. The generic node for the function MOL is examined in order to retrieve the calculation method of this function. The generic node returns the list of the concept nodes in which certain calculation methods are described. This list is called the WAITING-LIST for the subproblem (CALCULATE (O_2 MOL)). Because a calculation method appears in the set of knowledge attached to the concept CHEMICAL-MATERIAL, the WAITING-LIST looks like

(C_1 C_2, ...CHEMICAL-MATERIAL .....C_n)

where C_1, C_2, ... and C_n are concept nodes.

The problem-solver traverses the SORT-links from the concept

SODIUM to retrieve the super-concept of SODIUM. Whenever a concept node is encountered, the problem solver checks whether the same concept node is in the WAITING-LIST. If it is in the list, a calculation method of the property MOL is supposed to be there which can be applied to the current object O_2. The method will be activated. In the above example, the concept CHEMICAL-MATERIAL will be encountered and the concept node is in the WAITING-LIST. The retrieved method is

'If both the molecular-mass and the mass of a chemical-material are known, then the mol of the material can be calculated by dividing the mass by the molecular-mass.'

In the above example, the mass of O_2 is known to be 20 gr but the molecular mass of O_2 is unknown. So the subproblem (CALCULATE (O_2 MOLECULAR-MASS)) is created (see Fig. V-28 (b)).

The same process will be performed for solving this subproblem. The two methods for calculating the molecular-mass will be found in the knowledge about a chemical-material. One method uses the mass and the mol of the material, and the other uses the chemical-name of the material and the external data base TAB (see Fig. V-27). The first method cannot be applied to the current problem because the problem of finding the MOL of the material is a subproblem currently under consideration and may not appear twice in a subproblem chain. So the second method is applied. Because the method requires the chemical-name of the material, the subproblem (RETRIEVE (O_2 CHEMICAL-NAME)) is created (see Fig. V-28 (c)).

The declarations and calculation methods attached to the concept nodes and SELF nodes that were traversed are all applicable. Therefore, SODIUM is returned as the chemical-name of O_2. The subproblem (RETRIEVE (O_2 CHEMICAL-NAME)) can then be solved immediately. As a result, the problem (CALCULATE (O_2 MOLECULAR-MASS)) becomes solvable. The problem is solved by accessing the data base with the chemical-name = SODIUM. The result makes the next super-problem solvable and so on. Finally the whole problem becomes solved.

However, it should be noticed that there are usually several possible calculation methods for a certain problem and it is required to solve several subproblems before applying a certain method. In the above example there is only one possible method discovered for each subproblem and only one problem is created for each method. For general cases, the problem-subproblem hierarchy such as shown Fig. V-28 becomes an AND-OR tree.

V-8-3 Evaluation of Boolean Expressions

At Step 5 in the algorithm given in section V-8-1, the expressions attached to the SUBSORT-links, that is, the SELF-node conditions are evaluated in turn. The evaluation of the expressions results in either T or NIL. Therefore, we call the expressions boolean expressions and the predicates used in the expressions boolean predicates. We will describe here how to evaluate such expressions.

As described in section V-8-1, a concept node may be used in the descriptions attached to a SELF node. In this case, the concept node plays the role of a boolean predicate, that is, the expression ((CONCEPT NAME = WATER)(! SOLVENT)) in the example given in section V-8-1 is boolean. Because we discussed in detail how these expressions are treated in section V-8-1, we will explain the other predicates.

The predicate BETWEEN is a typical boolean predicate. We will see a simple example. A given description is as follows.

```
(CONCEPT      NAME = CHEMICAL-MATERIAL
   SUBSORT ; ((CONCEPT   NAME = LIQUID)
                 (BETWEEN  (!  BOILING-POINT)
                           (!  TEMPERATURE)
                           (!  CONGELATION-POINT)))
   ..................)
```

The description means

'If the temperature of a chemical material is between the boiling-point and the congelation-point, the material is liquid'. The condition for identifying a material with the concept LIQUID is expressed by using the predicate BETWEEN. The problem description is

(OBJECT-1 NAME = WATER
(! TEMPERATURE) = 15°C)

The object O_1 is first identified with the concept WATER. Here the problem is to prove that O_1 is liquid. Suppose the concept WATER is linked in the S.N. to the concept CHEMICAL-MATERIAL by a certain sequence of SORT-links, because CHEMICAL-MATERIAL is the common super concept of LIQUID and the object O_1 (see Fig. V-29).

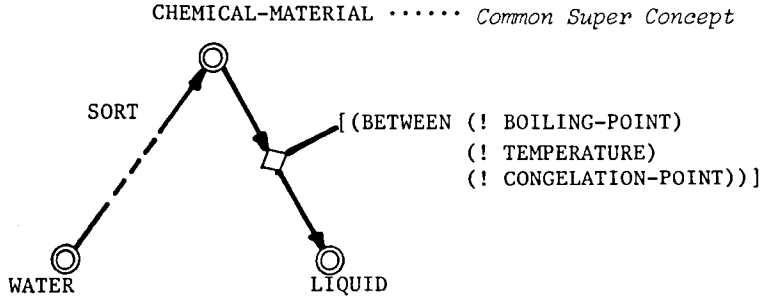


Fig. V-29 A Boolean Node in a SELF Description

In order to prove that O_1 is LIQUID we must prove the temperature of O_1 is between the boiling-point and the congelation-point. Before the evaluation of the expression, we must evaluate the path descriptions in the argument positions of BETWEEN. Therefore the problem hierarchy becomes the tree in Fig. V-30. The value of the first and the third path descriptions will be calculated by using the techniques described in section V-8-2. They may be retrieved from a certain external data base with CHEMICAL-NAME = WATER. The second subproblem can be solved simply by retrieving the value from the problem description.

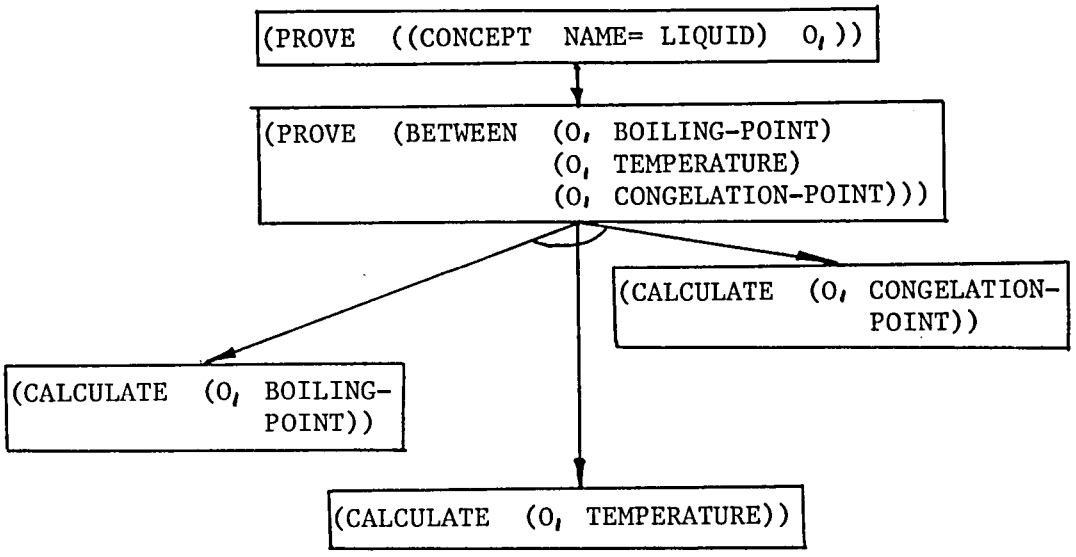


Fig. V-30 Problem-Subproblem Hierarchy

After these problems are solved, it is easy to determine whether the expression (BETWEEN (! BOILING-POINT) (! TEMPERATURE) (! CONGELATION-POINT)) is satisfied or not. This sequence of operations is controlled by the descriptions in the generic node of the predicate BETWEEN. Whenever BETWEEN is used as a boolean predicate, the above sequence of operations is performed. The descriptions in the generic nodes for BETWEEN and GREATER are roughly as follows.

```
[Description in the generic node for BETWEEN]
   make-subgoal[(PROVE (GREATER ARG1 ARG2));END]
   make-subgoal[(PROVE (GREATER ARG2 ARG3));END]
END return [NIL]

[Description in the generic node for GREATER]
   If TYPE(ARG1) = (NUMERIC, CONSTANT)
    then If TYPE(ARG2) = (NUMERIC, CONSTANT)
     then return [eval [greater [ARG1 ; ARG2 ]]
     otherwise  u = make-subgoal[(CALCULATE ARG2) ; END]
                return [eval [greater [ARG1 ; U]]]
    otherwise If TYPE(ARG2) = (NUMERIC, CONSTANT)
     then u = make-subgoal [(CALCULATE ARG1) : END]
```

```
            return [eval [greater [ U ; ARG2]]]
        otherwise u = make-subgoal [(CALCULATE ARG2) ; END]
                  v = make-subgoal [(CALCULATE ARG1) ; END]
                  return [eval [greater [ V ; U]]]
END     ..................
```

Notice that MAKE-SUBGOAL and TYPE are the LISP functions which are provided by the system for defining generic nodes. MAKE-SUBGOAL creates a subproblem which is specified by the first argument in the problem hierarchy. If the subproblem can be solved, the value of MAKE-SUBGOAL is the value which is returned as the solution to the subproblem. If the subproblem cannot be solved, the value of this function is NIL and the control will be transferred to the statements specified by the second argument.

The generic node for GREATER in the above description is somewhat simple. Sometimes certain subproblems cannot be solved but the expression can be easily evaluated. For example, suppose that the boiling-point of WATER is not given directly in the S.N. but only the following declarations are given.

```
(CONCEPT        NAME = WATER
    DECLARATION : (GREATER (!  BOILING-POINT)
                           ((CONCEPT NAME =ALCOHOL)  BOILING-
                                                      POINT)))

(CONCEPT        NAME = ALCOHOL
    DECLARATION : (GREATER (!  BOILING-POINT)  '80°C) )
```

Even in such a situation, human problem solver can solve the problem, because he can logically draw an inference by utilizing the properties of the predicate GREATER. In oder to do this in the S.N., we should describe the axioms about the predicate as the content of the generic node. It is easy to encode such programs in LISP. The properties of the predicates (BETWEEN, EQ, GREATER, LESS) which are built in the current version of the S.N. are rather simple. However there are various kinds of predicates which have more complex properties so that

it becomes difficult to encode them as LISP programs. Moreover because the axioms of several predicates intimately interact with each other, the programs in the generic nodes become intolerably complicated. We are now designing a new framework which makes it easy to encode the axioms about this type of predicate. In the current version of S.N. the only thing that we can do is to encode them by using a few LISP system functions like MAKE-SUBGOAL or TYPE. The current version of the description language is only capable of defining the properties of concept nodes in a descriptive form. There are many situations for which the system must be augmented. For example, one knows how to calculate the molecular-mass of a chemical compound from the chemical formula if he knows the molecular mass of each element of the compound. This knowledge should be represented as an arithmetic node. However, the definition of this arithmetic node cannot be expressed by a pure descriptive form.

V-9 Conclusion

We discussed in this chapter the problem of internal knowledge representation for our question-answering system. The semantic network S.N., we have proposed here, has many desirable characteristics. Some of them are listed in the following.

1. The description language for the S.N. can be seen as a programming language for problem-solving and knowledge representation in general. This programming language provides a powerful internal data structure in the form of the semantic network S.N..

2. The S.N. gives us useful frameworks for clustering relevant knowledge. Though recently developed A.I. languages have also the abilities of clustering relevant knowledge, the abilities are mostly based on the syntactic structures of expressions. Hash-

indexing used in PLANNER, CONNIVER and so on are typical examples of such index structures. On the contrary, the S.N. provides semantically meaningful index structures for relevant knowledge. Therefore, the process of retrieving relevant knowledge becomes itself a certain mode of reasoning.

3. The technique of pattern-directed function invocation is popular in recent A.I. languages. However, the pattern which is attached to a certain procedure does not necessarily reflect the content or the effect of the procedure. Because the structures of patterns do not have any logical implications, one can choose arbitrary pattern forms to express the same content. In the S.N. each function or procedure does not have any special external handles such as subroutine names, patterns and so on. On the contrary, each piece of knowledge is indexed by the content of the knowledge.

4. Various kinds of knowledge which are external to the network are naturally included in the S.N.. External data bases and arithmetic algorithms are examples. They are invoked when they are necessary during the problem solving process.

5. Because the representation of knowledge in the S.N. is highly descriptive, the same knowledge can be used for different purposes. For example, the statements which are attached to a SELF node can be used as both check conditions and declarations.

The current version of the S.N., however, is still insufficient in various aspects. Especially the following points are essential.

1. We must provide much more powerful and descriptive framework for defining new predicates in the S.N.. Certain kinds of knowledge can hardly be expressed in pure descriptive forms. It is not desirable to encode them directly by LISP, because the interactions between programs become intolerable for the designer to maintain. Some framework should be prepared to express them in more natural,

modular, and descriptive forms.

2. In the current version of the S.N., the local logical relationship can be properly represented. However, we cannot embed more global relationships between knowledge chunks in the S.N.. We should provide the ability for defining certain kinds of strategies or problem solving script in the S.N..

178項欠

CHAPTER VI

DICTIONARY ORGANIZATION AND MORPHOLOGICAL ANALYSIS

VI-1 Introduction

Nontrivial automatic processings on natural language texts invariably involve an operation called morphological analysis, in which the individual words in texts are recognized. We use the term, 'morphological analysis' which is somewhat more pretentious than the commonly used 'dictionary consultation', because we want to stress that there is more in recognizing words than just consulting a dictionary.

Most of the recently developed systems which attempt to analyze 'natural' language sentences can eventually treat only very limited subsets of the set of all natural sentences. First of all, these systems usually look on an input sentence as a sequence of words whose meanings are well defined in the dictionary. Syntactic and semantic analyses are directly applied to the input sentences.

The term 'word', in common usage, refers to a sequence of letters that is bounded by spaces or punctuation marks in text. According to this view, 'walk', 'walking', 'walks' and 'walked' are different words and therefore carry different meanings from each other. But common usage also allows all these to count as instances of a same lexical word, because they belong to the same lexical entry in an ordinary dictionary. According to this view, the above four words basically carry the same meaning, together with the additional information such as 'progressive', 'past' and so on. Hereafter, we will refer to a word which appears in texts and carries such additional information as a textual word, and a word which appears as headings in an ordinary dictionary as a lexical word.

Textual words are derived from underlying lexical words by morphological processes. From the computational point of view, we should be able to determine the corresponding lexical word from the given textual word. This restoration process is referred to as morphological analysis.

In English, textual words can be easily recognized, because they are usually separated from each other by spaces or punctuation marks in texts. However, there are many languages in the world in which there are no such separators, and textual words are simply juxtaposed. A sentence in these cases is just a sequence of letters at a glance. In these languages, therefore, the separation of each string into a sequence of textual words, and the determination of lexical words from textual words are very difficult tasks. Morphological analysis procedures for these types of languages must therefore be able to separate consecutive strings of letters into the units which are supposed to correspond to lexical entries and other interesting grammatical units. Japanese is one such language. Before applying morphological analysis to Japanese written texts, we must first extract 'textual' words from the texts.

In this chapter, we will describe a morphological analysis procedure for Japanese written texts together with a procedure which separates consecutive strings of letters into textual words. Because these procedures require dictionary consultation, we will also describe a storage structure for a large Japanese dictionary and methods for accessing it.

VI-2 Dictionary Organization

A dictionary is an important tool for language processing.

Various forms of dictionaries have been developed for mechanical processing of natural language. Most of them have been constructed tentatively in the course of developing experimental systems and intentionally adapted to specialized needs. The sizes of the vocabularies have been very limited. However, to establish techniques of language processing suitable for real applications such techniques should be validated for a large corpus of text data by testing them on a large dictionary.

Until recently there have been no attempts to computerize a large Japanese dictionary because of the complex writing system of the Japanese language. We have done extensive research on handling Japanese character strings by computer, and have computerized a large Japanese dictionary. We have developed convenient access methods for the dictionary in order to utilize it during the morphological analysis. The structure of this computerized Japanese dictionary has many interesting characteristics when considered as a data base. We shall explain these characteristics and the design of an efficient data structure for a Japanese dictionary in this section.

VI-2-1 The Computerized Dictionary

Algorithms for natural language processing must be verified by a large corpus of text data from various different fields. Therefore, the dictionary to be provided for the algorithms must necessarily be large enough to cover these different fields. But it is unthinkable in such a large dictionary to associate each word with detailed semantic descriptions which a computer can utilize efficiently. Therefore, we decided to computerize a conventional dictionary which is currently published and daily used. The dictionary we adopted is *'Shinmeikai Kokugojiten'* published by *Sansei-*

do Publishing Co., which contains about 70,000 lexical entries(see Table VI). The definitions of most words in the dictionary were very poor, and almost unusable by computer.

Part-of-Speech	Type of Inflection	Number of Entries
Transitive Verb	Type 1	88
Transitive Verb	Type 2	463
Transitive Verb	Type 3	1208
Intransitive Verb	Type 1	57
Intransitive Verb	Type 2	726
Intransitive Verb	Type 3	1351
Intransitive Verb	Type 4	271
Intransitive Verb	Type 5	12
Interjection		146
Prefix		51
Suffix		64
Adjective 1	Adjective Type 1	626
Adverb		1387
Pronoun		119
Noun		46279
Adjective 2	Adjective Type 2	1103
Adjective 2	Adjective Type 3	252
Nominalization Form of Verb		6760

* Postpositions, Inflectional Postpositions and Conjunctives are omitted from this table.

Table VI-1 Content of the Dictionary

The only information that could be utilized by computer were the spellings, the pronounciations, and the parts of speech of the lexical words.

VI-2-2 Data Structure for Japanese Dictionary

Entries in a Japanese dictionary are typical examples of multi-key records. That is, more than one key can be used to retrieve an entry. A word in Japanese usually has two different forms of spellings, a *Kana-spelling* and a *Kanji-spelling*. Because both spellings appear in ordinary texts, we need both of these be usable to identify entries in the dictionary. These are typical keys for the lexical entry. However, a key does not usually correspond to a single entry. It is often the case that there are several ectries with the same key value. One way of finding entries by using keys is to prepare index files for these key values. The access request for a Japanese dictionary has another characteristic that makes the data structure more complex. It is allowed to use several different *mixed spellings* of Kana-and Kanji-characters to represent the same word (see Fig. VI-1). It might prove necessary to use any of these spellings as keys to identify an entry during the morphological analysis of some particular text sample.

Kanji-Spelling	Kana-Spelling	Mixed Spelling	Meaning
大　学	だいがく	だ い 学 大 が く	University

Fig. VI-1 Example of the Kanji-, Kana-, and Mixed-Spellings

We decided to provide index structures for both the fundamental spellings, that is, Kana-and Kanji-spellings, and not to

provide any independent index structures for mixed spellings. A special complex search procedure is prepared for mixed spellings instead of index structures. When a mixed spelling is given, a procedure is activated to search the entries which have the specified mixed spellings. If the procedure were required to look up through the whole dictionary, the search time would be quite lengthy. In order to make the search easy, the index structures for Kana-spellings or Kanji-spellings are utilized to give the position for starting the search. The mixed spellings do not occur very often in a real text. To provide the dictionary system with the indices for mixed spellings greatly increases the quantity of secondary storage. Thus there is a trade off between the size of the dictionary storage and the time needed to locate the required entries.

In the sequel, we have the following keys for an entry in the dictionary.

A. *Fundamental keys*

1. Kana-spelling
2. Kanji-spelling

} These two types of keys are most frequently used, but they do not usually identify a unique word.

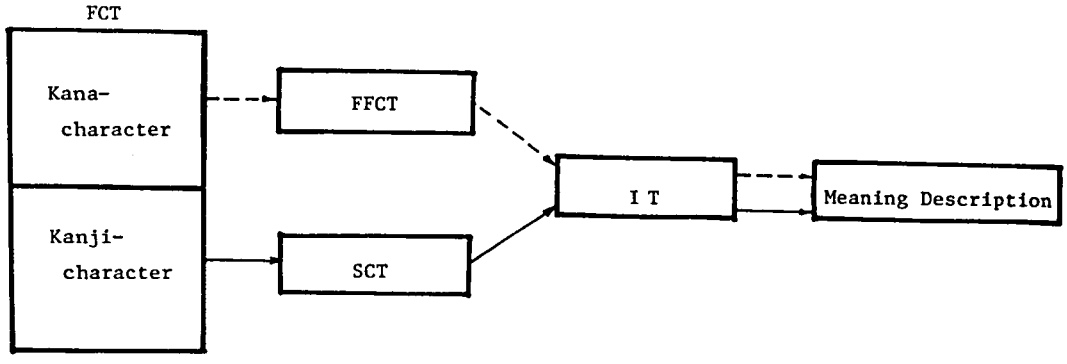
B. *Modified keys*

3. Mixed spelling whose first character is a Kana.
4. Mixed spelling whose first character is a Kanji.
5. Inflectional variant : This type of modified keys must first be transformed into lexical forms by the morphological analysis procedure using grammatical knowledge. The dictionary system provides only the procedure which identifies entries from the roots of the words or the words of their generic forms.
6. Variant of inflectional suffix : The inflectional suffixes

of some words are not unique. Fig. VI-2 shows such examples. Each one of these variants is stored as a lexical word in the dictionary and we can access it by simple means.

Kanji-Spelling	Kana-Spelling	Mixed Spelling	Meaning
繰 返 す	くりかえす	く り 返 す	to Repeat
繰り返す		繰 か え す	
		繰りかえす	

Fig. VI-2 Example of words which have several variants of inflectional suffixes



FCT : First Character Table

FFCT : First Five Characters Table

SCT : Second Kanji Character Table

I T : Item Table

Fig. VI-3 Overall Construction of the Dictionary System

Fig. VI-3 shows the overall construction of the dictionary system. The tables First Character Table (FCT), First Five Characters Table (FFCT), and Second Kanji Character Table (SCT), are all index tables. An entry in the item table (IT) contains information about the lexical word such as the spelling (both in Kanji and Kana), the part of speech, type of inflection if the word is inflectional, and so on. A pointer to the meaning description of the word is also included in the entry. The IT can also be viewed as one of the index tables for retrieving the meaning descriptions.

The tables, FCT, FFCT and IT form a multiple level index for Kana-spellings. On the other hand, the tables FCT and SCT form a multiple level index for Kanji-spellings.

1. FCT *(The first character table)* : This table consists of 4,096 entries each of which corresponds to a character (In our system, each character in Japanese is encoded by 12 bits). The index entry in FCT for a Kana-character will give the address of the head of the entries in FFCT, which begins with that character. The entry for a Kanji-character gives the address of the bucket in SCT for the character.

2. FFCT *(The first five characters table)* : An index entry in this table is prepared for every bucket in IT. A bucket in the IT contains 40 items, and the index entry in FFCT which contains the first five characters of the top key in that bucket is prepared. The entries in this table are sorted by the Kana-spellings. The reason why we restricted the key length of this table to the first five characters is that most of the lexical words in the dictionary are not longer than five characters (92.1%). Table VI-2 shows the distribution of the length of lexical entries in Kana-spellings.

3. SCT *(The second character table)* : The Kanji-spellings of the lexical words in the dictionary mostly consist of two Kanji-characters. The first Kanji-character is used as the first level index, that is, the entry of the FCT. The second Kanji-character

is stored in this table.

Length of Kana-Spelling	1	2	3	4	5	6	≥7
Accumulated Frequency	391	4217	17891	42449	53043	55932	57557
Relative Frequency (%)	0.6	7.3	31.0	73.7	92.1	97.1	100

Table VI-2 Distribution of the Length of Lexical Entries in Kana-Spellings

The entry in this table is a pair of the second character and the pointer to an item in IT. If there are two lexical items which have the same first two characters, two pairs whose first elements are the same (the second Kanji-character) but the second elements (pointers to the items in IT) are different are stored in this table.

4. IT *(Item table)* : The format of the item in this table is illustrated in Fig. VI-4. The items in this table are sorted by the Kana-spellings.

The relationship between these index tables is illustrated in Fig. VI-5. The storage requirements for each index table is given in Table VI-3.

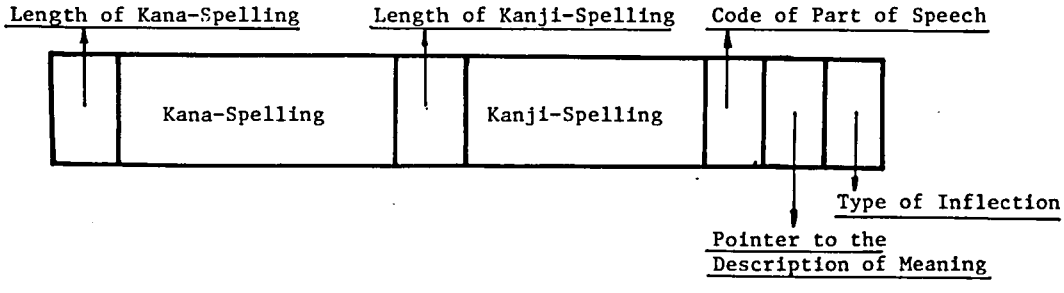


Fig. VI-4 Data Format of an Entry in IT

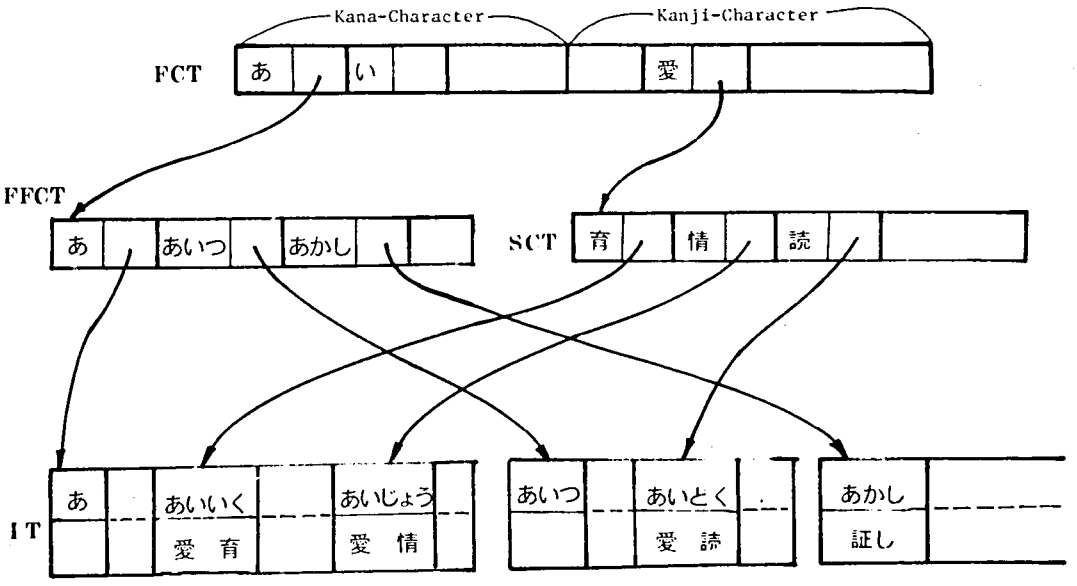


Fig. VI-5 Relationship between Index Tables

Index Table	Storage Requirement
FCT	24 KB
FFCT	18.6 KB
SCT	700 KB
I T	4.3 MB

Table VI-3 Storage Requirement for the Index Tables

VI-3 Morphological Analysis of Japanese Written Texts

Designers of speech understanding systems, who aimed at

interpreting consecutive phonetic symbols by using linguistic knowledge, have been faced with the problem of how to segment consecutive phonetic symbols into linguistically meaningful units before applying linguistic analysis on the input. In a natural utterance of sentences even in English, there are no stable clues for the extraction of textual words, which correspond to delimiters in written text. At the first stage of processing, the string of the input phonemes is roughly segmented into several disjoint parts. Each of the parts is enclosed with silent frames stably detected by preceding phonological processings. Each segmented part may contain more than one textual word because consecutive words are often pronounced together without any pauses.

In the morphological analysis of Japanese, we have been faced with the same problem. There are no spaces between textual words in written Japanese texts so that it is more difficult to extract textual words from written Japanese than from English texts. A great many characters are used in Japanese written texts (We can find about 2,000 - 3,000 different characters in ordinary texts). These characters are classified into two types. One is the type called Kana (Japanese phonetic character), and the other is the type called Kanji (Chinese character). Changes of character types as well as punctuation marks give us useful clues to boundaries where it is possible to separate a long character string (a sentence) into shorter, manageable units. A *'Pause group'* in Japanese grammar is a fundamental unit in a Japanese sentence, and contains only a single independent word (see section VI-3-1. Nouns, verbs, adjectives, adverbs and conjunctives are all independent words). Each unit obtained by detecting a change of character type may contain several pause groups, as the parts enclosed by silent frames in speech utterances may contain more than one textual word. It is necessary to segment the units further by consulting a dictionary. If we restrict our domain of discourse to a certain very limited field, the consultation of a dictionary and subsequent

morphological analysis would be very straightforward processes. However, when the vocabulary is very large, and the domain of discourse is not specifically limited, a lot of ambiguities may appear in the morphological analysis. In any case a computer-usable dictionary and a strong algorithm for the morphological analysis are necessary.

In this section, we will describe the way of segmentation of texts into pause groups, the way of utilizing the dictionary in the process, the strategy for the morphological analysis of Japaese written texts and the segmentation of long compounds of Chinese characters.

VI-3-1 Basic Characteristics of Japanese Texts

Before proceeding to the detailed description of morphological analysis of Japanese, we must make a brief sketch of the characteristics of Japanese which are relevant to the understanding of the following sections. The reader who is not familiar with Japanese may consult Kuno's work, (Kuno 1973), which explains some of the basic characteristics of Japanese in comparison with English.

First, we will have to explain the concept of *'Pause Group'*, PG for short, which plays the central role in the sentence construction of Japanese. In conventional grammar of Japanese, words are classified in two groups. One is the group of *independent words* which are nouns, verbs, adjectives, pronouns, adverbs, conjunctives and interjections. The other is the group of *dependent words* which are *postpositions* and *inflectional postpositions* (or auxiliary verbs). The distinction of independent words and dependent words roughly corresponds to the distinction of content words and function words. The unit we called PG consists of a single independent word

and several dependent words which follow the independent word. Fig. VI-6 shows the general construction of a PG.

$$(IW)\cdot(DW_1)\cdot(DW_2)\cdot\cdots\cdot(DW_n)$$

IW = Independent Word

DW_i = Dependent Word

The sequence of dependent words can be null.

Fig. VI-6 Typical Construction of a PG

As Kuno pointed out in (Kuno 1973), Japanese is a typical SOV language and, therefore, postpositional. In English, the case in surface structure is generally evident in the order of phrases. In Japanese a postposition attached to a noun phrase shows the case of the phrase in a sentence instead of ordering of phrases. A postposition in Japanese actually does more than simply plays a case marker. In a certain context the postposition, 'DAKE', when it is attached to a noun phrase, plays the role of the English adverb 'only', for example.

The underlined PG in the following is an example which contains more than one postposition.

彼(noun - he) こそ(postposition - emphasis of he) が(postposition - case marker of AGENT) 学校(noun - school) へ(postposition - case marker of DESTINATION) 行っ(verb - to go) た(inflectional postposition - past).

Meaning : It is he that went to school.

meaning of the postposition 'こそ'

In the above sentence, we have another postposition which has a different role from case markers, that is, the inflectional postposition 'た'. This postposition shows that the tense of the sentence is past or present perfect. The following sentence shows another example of the inflectional postpositions.

(1) 彼(noun - he) は(postposition - case marker of AGENT) 学校 (noun - school) へ(postposition - case marker of DESTINATION) 行か(verb - go, inflectional form 1) なかっ(inflectional postposition - negation, inflectional form 1) た(inflectional postposition - past).

meaning : He did not go to school.

(2) 彼 (noun - he) は(postposition - case marker of AGENT) 学校 (noun - school) へ(postposition - case marker of DESTINATION) 行か(verb - go, inflectional form 1) ない(inflectional postposition - negation, inflectional form 3) だろう(inflectional postposition - conjecture).

meaning : He may not go to school.

From the above two examples, one can recognize that a certain inflectional postposition requires that the preceding inflectional word should have a certain specific inflectional form. The postposition 'た'in Example (1) requires that the preceding textual word form be the inflectional form 1. On the contrary, the postposition 'だろう' in Example (2) requires the word of inflectional

form 3. Generally speaking, a (inflectional) postposition requires a certain condition that the immediately preceding word must satisfy. That is, a postposition restricts the part-of-speech and inflectional form of the preceding word. These conditions are checked during the morphological analysis procedure. We call the checking *'compatibility test'*.

Another thing that we must mention before proceeding to the detailed description of the morphological analysis procedure is the writing system of Japanese. The Japanese language has no definite delimiters of words such as spaces in English. How is it possible, therefore, to recognize the boundaries of words?

We have two types of characters in Japanese, Kanji-characters and Kana-characters. All (inflectional) postpositions are definitely written in Kana-characters. On the other hand, independent words are often written in Kanji-characters. Because an independent word comes in the left-most position in a PG, it can be expected that a PG begins from the position where the type of characters changes from Kana to Kanji. Though many independent words are written in Kanji-spellings, some independent words are written in Kana-spellings or in mixed-spellings. Even the same independent words are written in both Kana- and Kanji-spelling according to individual taste. So the attempt to detect PG boundaries is not always successful. In spite of this fact, the change of character types (from Kana to Kanji or vice versa) gives us useful clues for segmenting an input string into PG's.

VI-3-2 Morphological Analysis of Japanese

A procedure for morphological analysis of Japanese consists

of some or all of the following components :

1. *Detection of Boundaries between PG's*
2. *Analysis of Inflections*
3. *Dictionary look-up*
4. *Testing Compatibility of Words in PG's*
5. *Segmentation of Compound Words Composed of Kanji Characters*

We will discuss the first four components in this section. The problem of segmentation of long compounds will be discussed in the next section.

Morphological analysis is usually performed as the first step of the whole language processing task. Errors in this step can have damaging effects on the following processings. Our main objectives in morphological analysis are as follows :

1. All possible interpretations, instead of a single interpretation, must be given as the output of the processing.
2. We do not divide the analysis procedure into a definite sequence of distinct phases such as detection of PG boundaries, analysis of inflections, compatibility testing and so on. The components just described above are used intermixedly as the analysis procedes.
3. For the convenience of later improvement of the program, we carefully discriminate the rules against the general procedure that uses them. The discrimination makes it possible to improve the analysis program by changing only the rules, without having to change the procedure.
4. Idiomatic expressions are treated naturally in the program.

The analysis procedure operated as follows.

Step 1. *Detection of Boundaries between PG's*

Using the information of character types and punctuation marks, we first segment an input sentence into several substrings, each of which is expected to be a PG. We refer to these substrings as EPG's (expected PG's). In the following steps, the EPG's are the basic unit of the processing. See Fig. VI-7.

input sentence:

彼 は 実 験 を く り 返 し 行 っ た

correct segmentation:

彼 : noun - he
は : postposition - case marker of AGENT
実 験 : noun - experiment
を : postposition - case marker of OBJECT
く り 返 し : verb - to repeat, inflectional form 2
(This verb is written in the mixed spelling.)
行 っ : verb - to go, inflectional form 1
た : inflectional postposition - past

Extracted EPG's (After STEP 1)

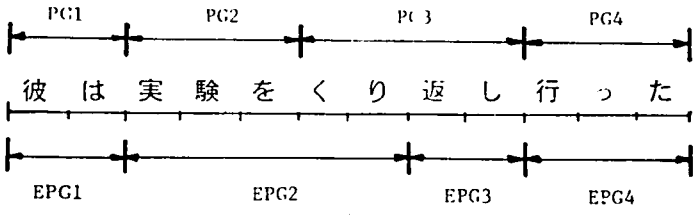


Fig. VI-7 Examples of EPG's

Step 2. *Interpretation of Kana-part*

General construction of an EPG created at Step 1 is shown

in Fig. VI-8.

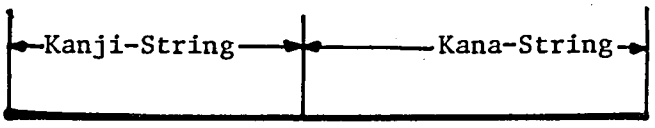


Fig. VI-8 General Construction of an EPG

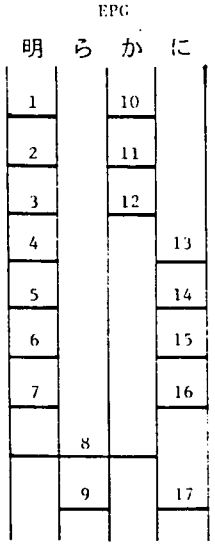
If the boundaries marked at Step 1 are correct, an EPG should have the construction of a PG as shown in Fig. VI-6. We assume tentatively that the boundaries are correct, and at this step we will decompose each EPG into lexical words and associate information from the dictionary with each word. As mentioned earlier, a textual word is not usually the same as a lexical word. Therefore, we must transform it to the lexical form before consulting the dictionary. In Japanese, each PG contains a single independent word. The remaining part of a PG is a sequence of dependent words. Every dependent word (postposition and inflectional postposition) is always written in Kana-characters, and the inlfectional suffixes of verbs, adjectives and other inflectional words are also written in Kana-characters. The stems of independent words are often written in Kanji-characters. So the Kana-part of an EPG consists of an inflectional suffix of an independent word if it is inflectional, and a sequence of dependent words.

We tabulated all inflectional suffixes, postpositions, inflectional postpositions, inflectional variants of irregular verbs, and the independent words which are usually written in Kana-spellings. These tables and their sizes are shown in Table VI-4.

Let us first assume that the Kanji-part in an EPG is a non-inflectional independent word (noun, adverb) or the stem of an inflectional independent word. Because consulting the dictionary of independent words is time-consuming, we do not refer the dic-

Table	Number of Entries
(Inflectional) Postposition	152
Independent Words usually written in Kana-Spelling	269
Inflectional Suffixes of Verbs	254
Inflectional Variants of Irregular Verbs	94
Independent Words which contain Kana-characters	60

Table VI-4 Sizes of the Tables for Inflectional Suffixes, (Inflectional) Postpositions, Inflectional Variants of Irregular Verbs and Independent Words which are usually written in Kana-spellings.



1. Noun
2. Stem of Adverb 1
3. Stem of Adverb 2
4. Stem of Verb, Inflection Type 1
5. Stem of Verb, Inflection Type 2
6. Stem of Verb, Inflection Type 3
7. Adverb
8. Stem of Adverb 2
9. Inflectional Suffix for Inflectional Form 1 of Verb (Inflection Type 1)
10. Inflectional Suffix for Inflectional Form 1 of Verb (Inflection Type 1)
11. Postposition (for Question)
12. Postposition (for Conjunction)
13. Inflectional Suffix for Inflectional Form 2 of Adjective 2
14. Inflectional Suffix for Inflectional Form 2 of Verb (Inflection Type 2)
15. Inflectional Suffix for Inflectional Form 2 of Verb (Inflection Type 1)
16. Inflectional Suffix for Inflectional Form 1 of Verb (Inflection Type 2)
17. Postposition (Case marker)

Fig. VI-9 Example of Interpretations of Substrings in an EPG

tionary to determine the interpretation of the Kanji-part of an EPG at this step. Instead, we will first interpret the Kana-part of an EPG by using the above tables. At this step, all possible interpretations of each substring, which may begin from arbitrary positions in the Kana-part are given. An example is shown in Fig.VI-9.

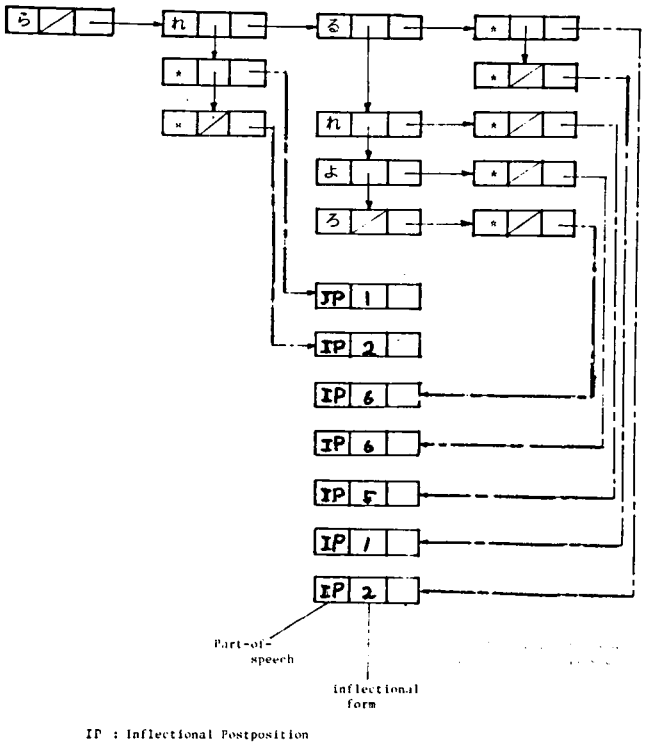


Fig. VI-10 Data Structure for the Table

The tables used at this step are represented by the tree-structures shown in Fig. VI-10 and are stored in the main memory. This is because the number of items which must be stored in these tables is relatively small and the consultation to these tables are very frequent. By using these tables, we can get all possible postpositions, inflectional postpositions, inflectional suffixes of independent words, and also independent words usually written

in Kana-spellings, which match with substrings starting at arbitrary positions in the input EPG.

Step 3 *Compatibility Testing*

As mentioned in section VI-3-1, a (inflectional) postposition has a certain condition that the immediately preceding word must satisfy. By using this condition, we can eliminate invalid interpretations of the EPG. Because most of the spellings of dependent words and inflectional suffixes consist of less than 3 Kana-characters, many possible interpretations of substrings are generated at Step 2. However, most of them are rejected by this compatibility testing. If the boundaries marked at Step 1 are not correct, there will be no interpretation which satisfies the whole EPG. When we cannot find any valid interpretation, we assume that the boundary detection was wrong, and will go to Step 4 to recover from the errors of the boundary detection. Otherwise, we will go to Step 5.

Step 4 *Recovery from Erroneous Boundary Detection*

The causes of errors which occur at Step 1 are classified into the following three types.

1. There is an independent word in the EPG which is written in Kana-characters. Though we have a table of the independent words which are usually written in Kana-spellings and use it in Step 2 to interpret EPG's,we have not listed the spellings of all independent words.
   In this case, the EPG consists of more than one PG (See 1, 2 in Fig. VI-11).

2. There is an independent word in mixed-spelling. In this case, the independent word is broken into two different

EPG's. If the first half of the word is written in Kana-characters, it is merged in the preceding EPG (see 3 in Fig. VI-11). If the second half of the word is in Kana-characters, we cannot interpret the Kana-part in the EPG at step 2 and 3 (see 4 in Fig. VI-11).

3. The preceding PG consists of only an independent word in Kanji-spelling. It is possible for a PG to have a null sequence of dependent words. We did not segment the string of Kanji-characters at the previous stages. Therefore we cannot detect that two independent PG's are merged (see 5 in Fig. VI-11). Because compounding in Japanese is very productive and several independent lexical words are concatenated to make a new compound, we will first assume a long string of Kanji-characters as a single compound. But in the example 5 in Fig. VI-11, the two consecutive independent words do not produce a compound and it is natural to think of it as the two independent PG's exist.

The errors of the third type were not discovered by the previous steps. They will be detected and recovered at Step 5 where a compound is segmented into primitive lexical words. Step 4 is responsible for recovery from the first two types of errors.

We assume therefore that the last Kana-character in the EPG eventually belongs to the succeeding PG, and delete the character from the current EPG. The remaining Kana-part will be re-interpreted by Steps 2 and 3. If the interpretation succeeds, then the independent word in the succeeding PG is assumed to be in the mixed-spelling or Kana-spelling and the deleted Kana-character is attached to the front of the next EPG. Then, the string created by the concatenation will be matched against lexical words at Step 5 (the way of matching an independent word in the mixed-spelling against lexical words has been discussed in section VI-2). If a lexical word which has a string of the mixed-spelling is found in

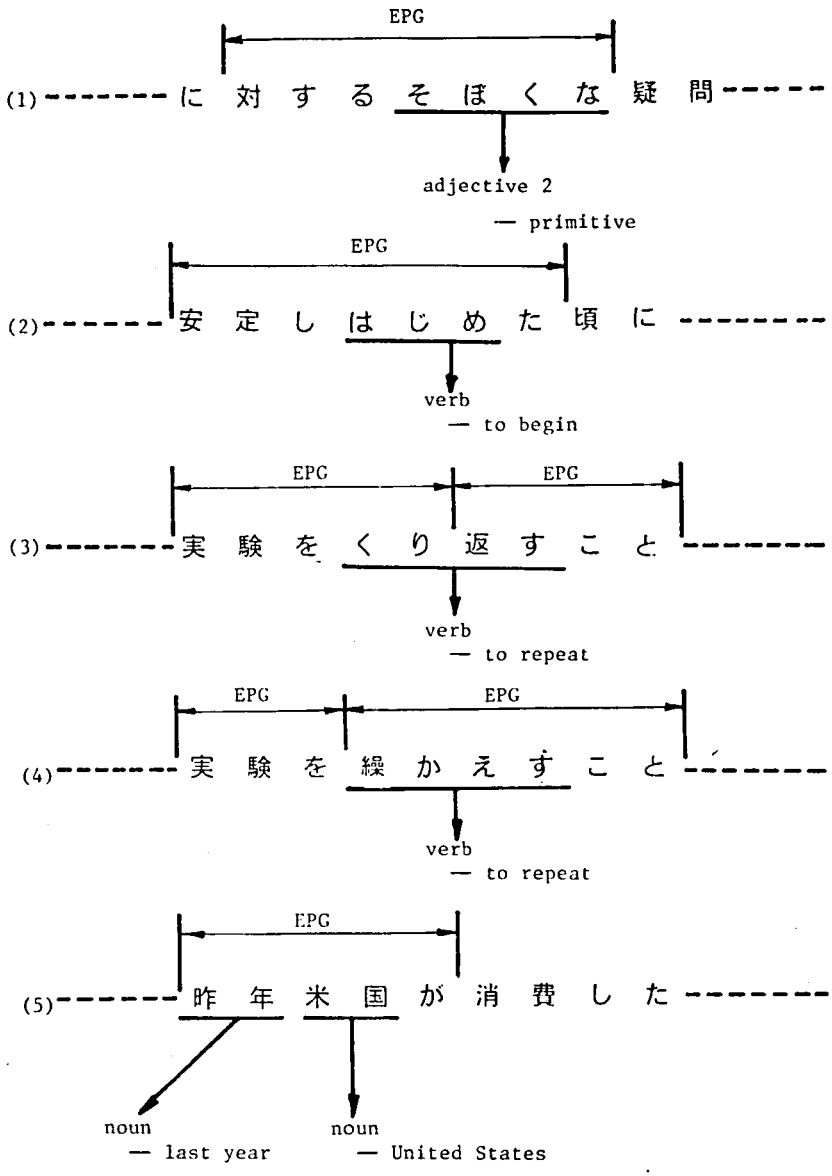


Fig. VI-11 Examples of Erroneous Boundary Detections

the dictionary, then the recovery from the error is completed. Otherwise, the last Kana-character is deleted again from the current EPG and the same steps will be repeated until the re-interpretation succeeds or the remaining part contains no Kana-characters. If there are no Kana-characters to be deleted and valid

interpretation has not been found, error cannot be recovered and the EPG has no interpretations.

Step 5 *Dictionary Look-up*

Because consulting the dictionary is very time consuming, it is desirable that the consultation is performed as rarely as possible. Moreover, the dictionary of independent words is never complete, that is, most of vocabulary in a specific domain such as the chemical field, electrical engineering, and so on are usually not in an ordinary dictionary. The same is true for most of compounds which often appear in a real text, but which are not regarded real words. On the other hand, it is possible to list all the dependent words and inflectional suffixes in the tables. Therefore, we interpret the Kana-part before the interpretation of the Kanji-part.

Several possible interpretations of the Kana-part may be produced by the above processing. Each interpretation makes certain assumptions about the Kanji-part in the EPG. For example, if an interpretation of the Kana-part begins with the inflectional suffix of a verb,then the interpretation of the Kana-part assumes that the Kanji-part is a stem of a verb. In this case, the generic form of the verb can be restored from the suffix information. Therefore, it is an easy task to check whether the assumption is fulfilled by looking up the verb in the dictionary.

The interpretation whose assumption about the Kanji-part is fulfilled by the dictionary is more feasible than the interpretations whose assumptions are not fulfilled. An example is given in Fig. VI-12.

However, if the Kanji-part is not a single lexical word but a compound which contains several lexical words, the process of consulting the dictionary becomes much more complicated. The detailed process of segmentation of compounds will be described in

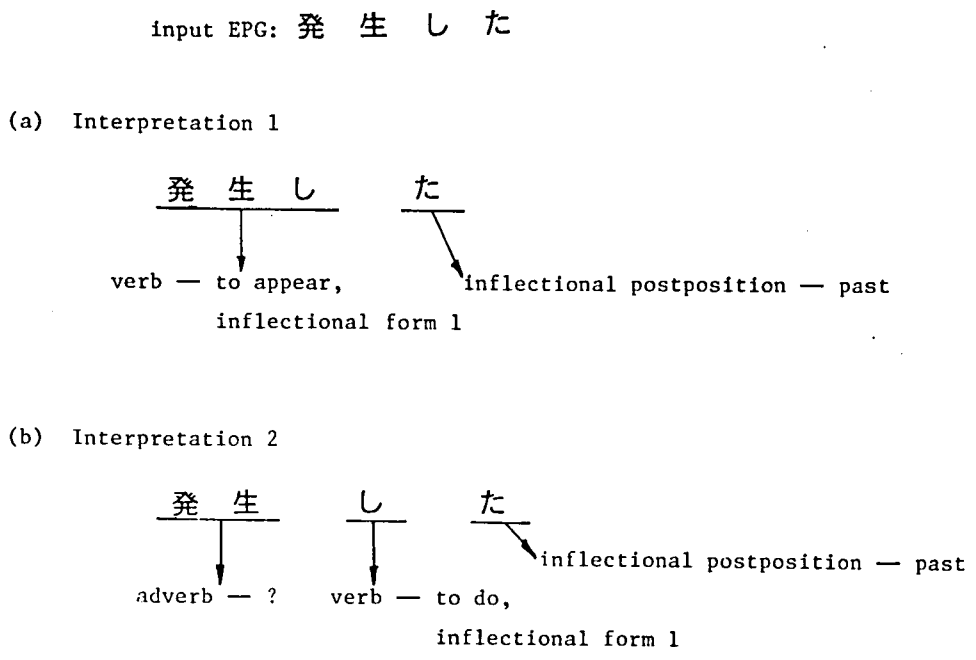


Fig. VI-12 The above two interpretations are equally feasible before consulting dictionary because both of them pass the compatibility test. However, consultation of dictionary indicates that the first interpretation is more feasible than the second.

section VI-4. The procedure for segmentation of compounds will also find the errors of the third type which are mentioned at Step 3.

Even if the independent word of an EPG is not found in the dictionary, the processing of the EPG does not fail. In this case, the interpretations proposed by the previous steps are considered as equally feasible. We consult the dictionary only to order the interpretations in terms of feasibility.

VI-3-3 Experimental Result of Morphological Analysis

Table VI-5 shows the results of the morphological analysis procedure when it is applied to the sentences from an elementary chemistry text book. Some analysis results are shown in Fig. VI-13. The processing time is also shown in Table VI-5. The computer system which we utilized for this experiment is shown in Fig. VI-14. The mini-computer has a 64 KB core memory. The output device of Kanji and Kana characters is a dot printer by which each character is expressed by 24 x 24 dots. We have dot patterns for about 2,000 characters, which is sufficient for this experiment. Most of the programs were written in FORTRAN and the input and output programs for Kanji and Kana characters were written in assembler language.

Correct Rate of Analysis (%)

	Single Correct Result		Plural Results are obtained. Correct one is contained.		Failure	
	Word	Pause Group	Word	Pause Group	Word	Pause Group
Before Consultation of Dictionary	73.0	61.4	20.9	33.2	6.1	5.4
After Consultation of Dictionary	87.0	74.8	9.5	20.0	3.5	5.2

(*) 500 PG's are processed by the procedure.

Average Processing Time (second)

	per a sentence	per a PG	per a word
Without Dictionary Consultation	4.6	0.43	0.17
With Dictionary Consultation	35.3	3.32	1.30

(*) The processing time include the times for MT I/O.

Table VI-5 Result of Morphological Analysis

すぐれた観察者は、これらの疑問をたいせつにして思考を積み重ね、正しい知識を得ようと努力するものである
。

Analysis Result :

```
*
*すぐれ（動下用）た（助動体）／（３）／*
*
*観察者（名　　）は（副助　）、（読点　）／（０）／*
*
*これ（代名　）ら（接尾　）の（格助　）／（１）／*
*
*疑問（名　　）を（格助　）たいせつに（形動用）し（動サ用）て（接助　）／（０）／*
*
*思考（名　　）を（格助　）／（１）／*
*
*積（副　　）み（動上用）／（０）／
*積み（動上用）／（０）／
*積み（動五用）／（１）／*
*
*重（名　　）ね（終助　）、（読点　）／（０）／
*重ね（名　　）、（読点　）／（０）／
*重ね（動下用）、（読点　）／（１）／
*重ね（動五命）、（読点　）／（０）／
*重ね（動五仮）、（読点　）／（０）／*
*
*正（名　　）し（動サ用）い（動上用）／（１）／
*正（副　　）し（動サ用）い（動上用）／（０）／
*正し（動五用）い（動上用）／（１）／
*正しい（動五用）／（０）／
*正しい（形容体）／（１）／*
*
*知識（名　　）を（格助　）／（１）／*
*
*得（動下未）よう（助動終）と（接助　）／（１）／*
*
*努力（名　　）する（動サ体）もの（形名　）で（助動用）ある（動五終）。（句点　）／（１）／
*努力（名　　）する（動サ体）もの（形名　）で（格助　）ある（動五終）。（句点　）／（１）／
```

The part-of-speech and the inflectional form of each character string are given in ( ) which follows the string.

Fig. VI-13 Examples of Analysis Results -- Example 1

ブラウン運動は、水の分子が花粉などの微細な粒子に衝突するために起こると考えるとうまく説明ができるので、分子の存在、またはその運動を証明する１つの有力な現象とみなされている。

Analysis Result :

```
*
*ブラウン運動（名　　）は（副助　）、（読点　）／（０）／*
*
*水（名　　）の（格助　）／（１）／*
*
*分子（名　　）が（格助　）／（１）／*
*
*花粉（名　　）など（副助　）の（格助　）／（１）／
*花粉な（形動体）どの（連体　）／（０）／*
*
*微細な（形動体）／（０）／*
   •
   •
   •
   •
*
*有力な（形動体）／（０）／*
*
*現象（名　　）と（格助　）みな（名　　）さ（動サ未）れ（助動用）て（接助　）いる（動上終）。（句点
  ）／（３）／
*現象（名　　）と（格助　）みな（副　　）さ（動サ未）れ（助動用）て（接助　）いる（動上終）。（句点
  ）／（２）／
*現象（名　　）と（格助　）みなさ（動五未）れ（助動用）て（接助　）いる（動上終）。（句点　）／（３
）／*
```

This substring was initially assumed to be a sequence of independent words, because it is written in Kana-characters. However, the compatibility test at Step 3 rejected this assumption. The substring was restored at Step 4 to be a verb in the Kana-spelling.

Fig. VI-13 (Continued) -- Example 2

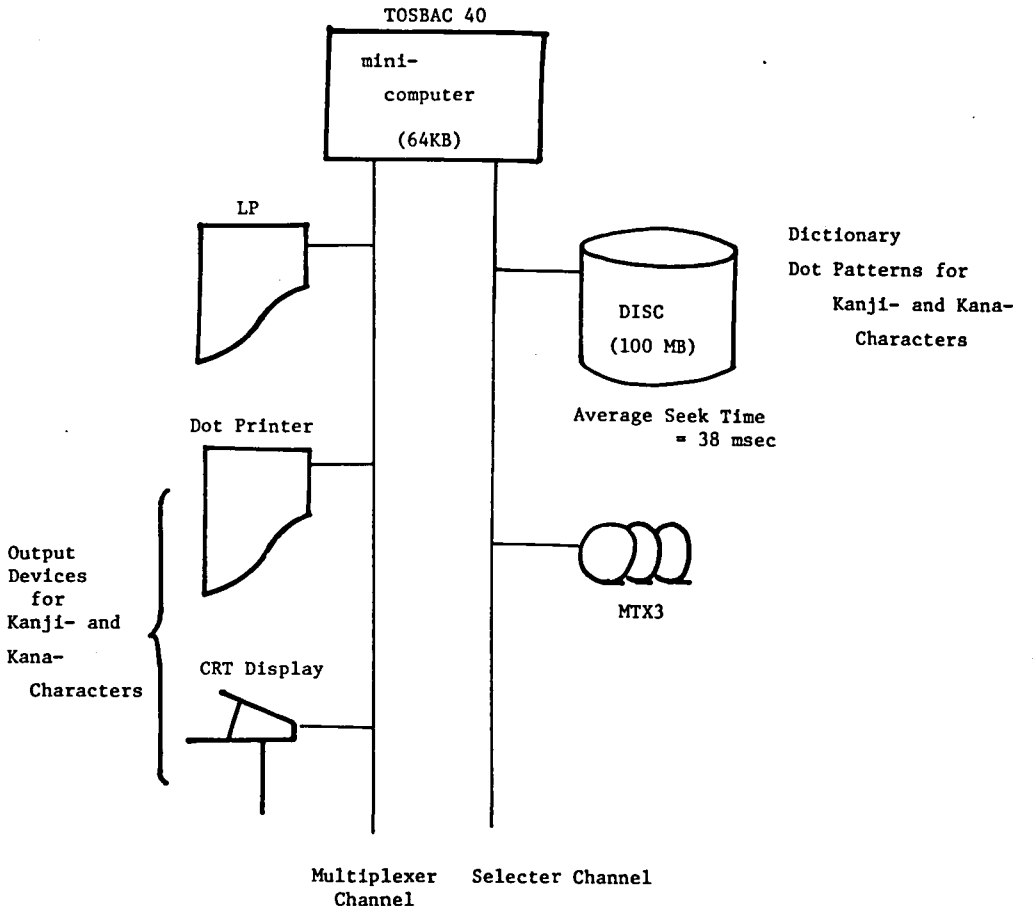


Fig. VI-14 Computer System

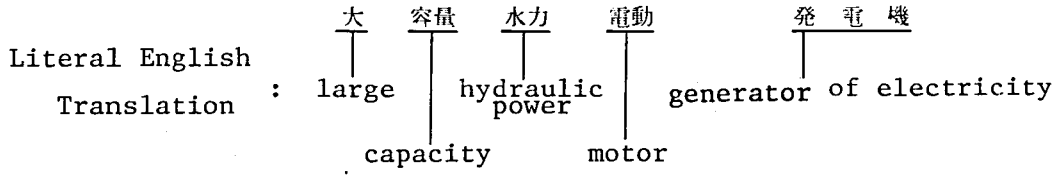
If there are several possible interpretations, the procedure outputs all of them. The PG underlined in Example 1 has two alternative interpretations. Example 2 illustrates that an embedded independent word in an EPG is well recognized by the error recovery procedure at Step 4.

Some kinds of ambiguities cannot be resolved by morphological rules alone. Syntactic, semantic and even pragmatic analyses would be required to resolve them. Certain simple heuristic criteria may be conceivable for selecting the most feasible interpretation. One such criterion is to select the interpretation which

contains the longest independent word. However, as mentioned at the beginning of this section, we prefer obtaining all possible interpretations to selecting the most feasible one at this stage. Disambiguation will be done at the succeeding stages such as syntactic, semantic, and pragmatic analyses.

VI-4 Compound Noun in Japanese

In the past many systems of automatic processing of English text intentionally avoided the problem of inflectional variants, and simply declared each of them to be an independent lexical word. This has an apparent advantage of eliminating the distinction between textual and lexical words and the morphological analysis reduces to a simple process of looking up forms in a dictionary. However, this solution is unfeasible in languages like Japanese and German, because compounding of words, which is very productive in these and many other languages, makes it impossible to consult a dictionary in a simple straight manner. There are lots of compounds in Japanese such as,



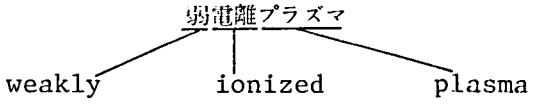
Compounds of a few words which are composed of more than 10 Kanji-characters are by no means extra-ordinary in Japanese. Many noun phrases, which must be written using prepositional phrases, infinitive phrases and so on in English, can be represented by single compounds of Kanji-characters in Japanese. Therefore, if we list

up all possible textual words in a dictionary, the size of the dictionary would be incredibly large.

The morphological analysis procedure of Japanese must be provided with the ability of separating a compound textual word into lexical words. We shall discuss this problem in the following sections.

VI-4-1 Segmentation of Japanese Compounds

A phrase in English, '*weakly ionized plasma*' corresponds to a single compound in Japanese,



At Step 5 in the morphological analysis procedure, we consult the dictionary to find the lexical entry of 弱電離プラズマ. This compound is probably not listed in the dictionary. Before consulting the dictionary, we should segment a compound into elementary lexical entries. Moreover, we should be able not only to segment compounds into lexical entries but also to discover the relationship among them, if we aim at translating Japanese into another language. That is, if we want to translate the compound : '弱 (*weakly* or *weak*) 電離 (*to be ionized* or *to ionize*) プラズマ (*plasma*)' correctly, we must select the structure of elements which means '*plasma which is weakly ionized*' from other possible structures corresponding to the meaning : '*plasma which ionize weakly*', '*weak plasma which is ionized*' and so on.

The problem of selecting the proper structure is closely related to the problem of semantics and pragmatics. The analysis of a long noun phrase concatenated by the particle 'NO' (of) is

relevant to this problem. We described in Chapter III how semantic and pragmatic knowledge interplay with each other to relate each component word of a phrase without syntactic clues. We will concentrate in this section on how to segment a compound into elementary lexical words.

'*Japan Information Center of Science and Technology*', JICST for short, delivers a quick report of current scientific papers twice a month, called '*Current Bibliography on Science and Technology*'. We used this as the data for investigating the nature of Japanese compounds. This bibliography contains not only the bibliographic information, but also the abstract of each paper in Japanese language. We used the materials published from June through November 1975 as our corpus for experiment.

We extracted all the strings in the abstracts which consisted of consecutive Kanji-characters. We gathered 74,127 different Kanji-strings from the corpus. As the extraction was done only by finding the change of character types (Kana to Kanji, etc.), there were some strings which are not single compounds, but are composed of two completely independent words.

(%)

Length	3	4	5	6	7	8	9	10	11	12	13
JICST	17.6	34.8	20.5	14.2	6.7	3.4	1.5	0.6	0.3	0.2	0.2
Newspapers	47.8	34.9	10.1	4.4	1.9	0.7	0.1	0.1	0	0	0

JICST : Compounds extracted from the abstracts in 'Current Bibliography on Science and Technology'. 68,021 different Kanji-character strings are examined.

Newspaper : Compounds extracted from newspapers. 1505 different strings are examined.

Table VI-6 Distribution of the Length of the Compounds

However, most of the extracted strings (98% were single compounds. Table VI-6 shows the distribution of the length of the compounds. We can easily recognize that the vocabularies in scientific papers tend to be longer than those in ordinary newspapers and magazines. This is because technical terms usually express more specific and complex concepts than those in normal texts. In English, a specific and complex concept is often expressed by a phrase in which a fundamental concept is modified by several adjective and prepositional phrases. In Japanese, this is expressed by a long compound. Therefore, in order to apply linguistic techniques to the areas of information retrieval, automatic abstraction and so on, we should be able to deal with these long compounds.

Most Japanese nouns are expressed by concatenating two Kanji-characters. As a result compound nouns can be easily segmented by separating them into two-character components. However there are some Kanji-characters which have definite meaning of their own. The characters, 熱 *(heat)*, 光 *(light)*, 国 *(nation)* are such examples. There are the other types of characters which have certain functions by themselves such as prefixes and suffixes. '大' *(large, big)*, 非 (prefix of negation) are typical examples of prefixes and 的 (suffix which makes an adjective), 化 (suffix which makes a noun) are typical examples of suffixes.

Fig. VI-15 shows examples of correctly segmented compounds. The underlined characters are prefixes or suffixes.

The existence of these special characters gives us useful clues to segment a Kanji-string. But it is not always true that these characters are used as prefix or suffix. Very often they are used in ordinary nouns. For example, all of the following Kanji-strings begin with the character '大' which are often used as a prefix to mean *'large'*, *'heavy'*, *'big'* or *'great'*.

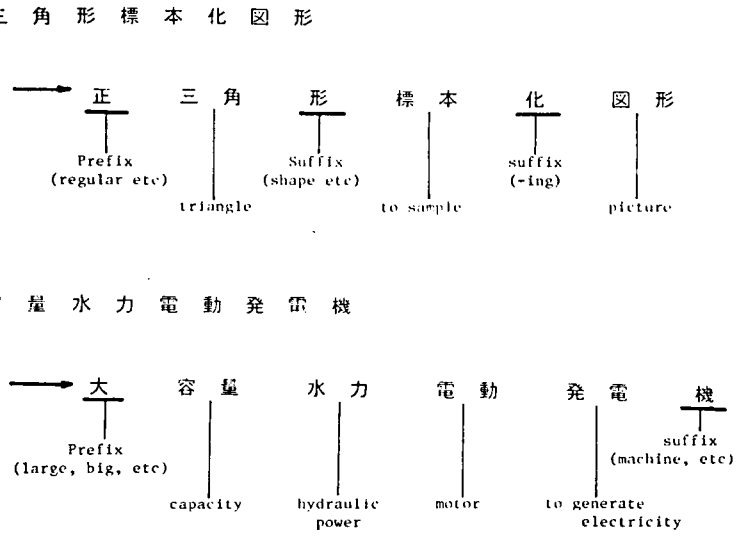


Fig. VI-15 Examples of Correctly Segmented Compounds

大雨	heavy rain
大男	big man
大蔵	finance ministry
大君	emperor
大詰	final scene
大様	generous

However, the character is used as a prefix only in the first two examples. The character in the other strings is used in the spelling of ordinary nouns. We can reconstruct the whole meaning of 大男*(big man)* from the meanings of 大 *(big, large,* etc.) and 男 *(man)*. On the contrary, we cannot reconstruct the whole meaning of 大蔵 *(finance ministry)* from the meanings of 大 and 蔵*(warehouse, storehouse)*. It may be possible but very difficult for a computer program to suppose the meaning of 大君 *(emperor)* from the meanings

of 大 (*big, large, great,* etc.) and 君 (*king,* etc.). Though each Kanji-character originally has its own meaning (as in Chinese), and therefore, the meaning of a word consisting of several characters can theoretically be reconstructed from the meanings of the characters, we treat a string of Kanji-characters as a lexical entry if it conveys a certain concrete and indivisible meaning. The length of such Kanji-strings is usually two. The above example shows that the Kanji-characters which are used as prefixes and suffixes in Japanese cannot be clearly discriminated from other ordinary Kanji-characters. Every Kanji-character, even the ones which have a strong tendency to be used as suffixes or prefixes, occurs in the spelling of ordinary lexical words.

A string of three Kanji-characters has the construction of either a prefix followed by a lexical word or a lexical word followed by a suffix. If either the first two characters or the last two characters of a string is matched with a lexical entry in the dictionary (the dictionary which is used here is that described in Section VI-2), and not the both simultaneously, we can almost safely segment the string into 2-1 or 1-2 respectively (we applied this algorithm on 1,632 strings which satisfy the above condition, and the results were compared with the manually segmented results. The algorithm gives the correct segmentation for 1,503 strings(98.2%)). The remaining one character is supposed to be used as a prefix or suffix. Based on this assumption, we measured automatically the frequency that a certain character is used as a prefix or a suffix. The Table VI-7 shows the result which is obtained by applying the above method to 11,963 strings. The frequency in this table reflects to some extent the characteristic of each character, that is, how often the character is used in the spelling of ordinary lexical words, and how often it is used as a prefix or a suffix. We will refer this table as Prefix and Suffix Table (PST for short) in the following. The PST will be utilized in the process of the segmentation of long compounds described in the next section.

Prefix Table

課 = 1 0 0 / 0 0 0 1	諸 = 0 9 8 / 0 0 6 7	非 = 0 9 7 / 0 0 4.4
各 = 0 9 5 / 0 1 8 6	他 = 0 9 0 / 0 0 3 1	助 = 0 8 5 / 0 0 0 7
月 = 0 8 0 / 0 0 0 5	双 = 0 8 0 / 0 0 0 5	右 = 0 8 0 / 0 0 0 5
逆 = 0 7 8 / 0 0 3 8	新 = 0 7 6 / 0 0 4 3	旧 = 0 7 5 / 0 0 0 4
弱 = 0 7 2 / 0 0 1 1	両 = 0 7 2 / 0 0 5 1	主 = 0 7 0 / 0 0 5 5
横 = 0 6 9 / 0 0 1 3	恒 = 0 6 6 / 0 0 0 3	親 = 0 6 6 / 0 0 0 3
縦 = 0 6 6 / 0 0 1 2	全 = 0 6 6 / 0 1 0 9	疎 = 0 6 6 / 0 0 0 3
低 = 0 6 5 / 0 0 7 8	無 = 0 6 4 / 0 0 5 0	再 = 0 6 3 / 0 0 4.9
枝 = 0 6 2 / 0 0 0 8	零 = 0 6 2 / 0 0 2 4	悪 = 0 6 0 / 0 0 0 5
熱 = 0 6 0 / 0 1 0 7	本 = 0 6 0 / 0 0 7 8	核 = 0 5 8 / 0 0 1 7
何 = 0 5 7 / 0 0 0 7	米 = 0 5 7 / 0 0 0 7	四 = 0 5 4 / 0 0 1 1
総 = 0 5 3 / 0 0 1 5	断 = 0 5 3 / 0 0 6 4	雷 = 0 5 3 / 0 0 1 3
高 = 0 5 1 / 0 0 9 6	引 = 0 5 0 / 0 0 0 8	国 = 0 5 0 / 0 0 0 2
丸 = 0 5 0 / 0 0 0 2	渦 = 0 5 0 / 0 0 0 2	銀 = 0 5 0 / 0 0 1 0
句 = 0 5 0 / 0 0 0 2	剛 = 0 5 0 / 0 0 0 2	小 = 0 5 0 / 0 0 6 1
床 = 0 5 0 / 0 0 0 6	週 = 0 5 0 / 0 0 0 2	青 = 0 5 0 / 0 0 0 2
待 = 0 5 0 / 0 0 0 6	田 = 0 5 0 / 0 0 0 2	働 = 0 5 0 / 0 0 0 4
脳 = 0 5 0 / 0 0 0 6	励 = 0 5 0 / 0 0 0 2	同 = 0 4.5 / 0 0 6 0
未 = 0 4.5 / 0 0 1 1	多 = 0 4 3 / 0 0 4.6	巻 = 0 4 2 / 0 0 0 7
半 = 0 4.2 / 0 0 2 8	的 = 0 4 1 / 0 0 1 7	鋼 = 0 4.1 / 0 0 1 7
二 = 0 4 1 / 0 0 5 8	英 = 0 4 0 / 0 0 0 5	既 = 0 4.0 / 0 0 0 5
今 = 0 4 0 / 0 0 0 5	苗 = 0 4 0 / 0 0 0 5	信 = 0 4 0 / 0 0 1 0
暗 = 0 3 8 / 0 0 1 3	筋 = 0 3 8 / 0 0 1 3	静 = 0 3 8 / 0 0 2 1
偏 = 0 3 8 / 0 0 1 8	油 = 0 3 8 / 0 0 1 3	軟 = 0 3 7 / 0 0 0 8
耐 = 0 3 6 / 0 0 1 9	頭 = 0 3 6 / 0 0 1 1	際 = 0 3 5 / 0 0 1 4
複 = 0 3 5 / 0 0 2 0	光 = 0 3 4 / 0 1 8 0	閉 = 0 3 4 / 0 0 2 6

Relative frequency (%) in which the character is used as a prefix

A = number / number

Total frequency of the character

Table VI-7 Prefix and Suffix Tables

Suffix Table

者 = 0 8 9 ／ 0 0 9 3	例 = 0 8 9 ／ 0 0 9 6	棒 = 0 8 8 ／ 0 0 1 8
系 = 0 8 7 ／ 0 2 1 0	型 = 0 8 7 ／ 0 0 8 9	剤 = 0 8 5 ／ 0 0 2 1
側 = 0 8 5 ／ 0 0 6 1	法 = 0 8 4 ／ 0 2 7 8	炉 = 0 8 2 ／ 0 0 2 8
器 = 0 8 1 ／ 0 1 8 1	鏡 = 0 8 1 ／ 0 0 2 2	等 = 0 8 1 ／ 0 1 3 3
盤 = 0 8 1 ／ 0 0 1 6	片 = 0 8 1 ／ 0 0 1 6	貝 = 0 8 0 ／ 0 0 1 0
厚 = 0 8 0 ／ 0 0 0 5	室 = 0 8 0 ／ 0 0 4 2	台 = 0 8 0 ／ 0 0 2 0
窓 = 0 7 8 ／ 0 0 1 4	値 = 0 7 8 ／ 0 1 8 5	塔 = 0 7 7 ／ 0 0 0 9
綱 = 0 7 7 ／ 0 0 6 2	群 = 0 7 6 ／ 0 0 4 2	図 = 0 7 5 ／ 0 0 0 4
後 = 0 7 5 ／ 0 0 8 3	店 = 0 7 5 ／ 0 0 0 4	粉 = 0 7 5 ／ 0 0 0 4
弁 = 0 7 5 ／ 0 0 2 4	上 = 0 7 4 ／ 0 2 0 8	図 = 0 7 4 ／ 0 0 7 5
部 = 0 7 4 ／ 0 2 2 4	則 = 0 7 3 ／ 0 0 3 4	府 = 0 7 3 ／ 0 0 9 1
率 = 0 7 3 ／ 0 1 3 3	局 = 0 7 2 ／ 0 0 5 1	項 = 0 7 2 ／ 0 0 3 7
性 = 0 7 2 ／ 0 4 2 1	板 = 0 7 2 ／ 0 0 6 6	案 = 0 7 1 ／ 0 0 1 4
所 = 0 7 1 ／ 0 0 4 2	化 = 0 7 0 ／ 0 2 2 1	軸 = 0 6 9 ／ 0 0 4 2
省 = 0 6 9 ／ 0 0 1 3	膜 = 0 6 9 ／ 0 0 3 9	式 = 0 6 8 ／ 0 1 6 6
官 = 0 6 6 ／ 0 0 0 3	額 = 0 6 6 ／ 0 0 0 6	広 = 0 6 6 ／ 0 0 0 6
個 = 0 6 6 ／ 0 0 0 9	湖 = 0 6 6 ／ 0 0 0 3	財 = 0 6 6 ／ 0 0 0 3
士 = 0 6 6 ／ 0 0 0 3	版 = 0 6 6 ／ 0 0 0 3	薬 = 0 6 6 ／ 0 0 0 3
翼 = 0 6 6 ／ 0 0 0 3	比 = 0 6 5 ／ 0 0 8 2	城 = 0 6 4 ／ 0 0 9 8
車 = 0 6 4 ／ 0 0 2 8	紙 = 0 6 4 ／ 0 0 1 4	角 = 0 6 3 ／ 0 0 6 0
内 = 0 6 3 ／ 0 1 1 9	帯 = 0 6 2 ／ 0 0 4 5	物 = 0 6 2 ／ 0 0 9 3
量 = 0 6 2 ／ 0 2 1 7	源 = 0 6 1 ／ 0 0 8 1	中 = 0 6 1 ／ 0 2 4 5
点 = 0 6 1 ／ 0 1 8 6	費 = 0 6 1 ／ 0 0 4 9	用 = 0 6 1 ／ 0 2 7 0
雲 = 0 6 0 ／ 0 0 0 5	街 = 0 6 0 ／ 0 0 0 5	孔 = 0 6 0 ／ 0 0 1 5
川 = 0 6 0 ／ 0 0 0 5	卓 = 0 6 0 ／ 0 0 0 5	時 = 0 5 9 ／ 0 1 9 9
壁 = 0 5 9 ／ 0 0 2 7	家 = 0 5 8 ／ 0 0 1 7	前 = 0 5 8 ／ 0 0 3 6

Relative frequency (%) in which the character is used as a suffix

A = number / number

Total frequency of the character

Table VI-7 (Continued)

VI-4-2 Segmentation Procedure of Long Compounds (1)
Procedure without Dictionary Consultation

Based on the PST, we manually classified Kanji-characters into the following five categories :

1. *Prefix characters* (PC) : The characters which are often used as prefixes and not used as suffixes.
2. *Suffix characters* -1 (SC-1) : The character which are often used as suffixes but not used as prefixes.
3. *Suffix characters* -2 (SC-2) : The same as above 2 except that the string which has this type of suffix becomes an adjective.
4. *Independent characters* (IC) : The characters which are often used both as suffixes and prefixes. These characters convey definite meanings by themselves.
5. *Ordinary characters* (OC) : The character which are rarely used as suffixes and prefixes.

Examples of these categories are tabulated in Table VI-8.

Category	Examples
PC	諸、非、各、双、逆、新、旧、弱、両 ……
SC-1	者、例、系、剤、法、炉、器、等、盤 ……
SC-2	的、型、形、状 ……
IC	光、熱、水、薬、家、銅、鉄、海 ……
OC	影、重、坑、結、気、検、消、速 ……

Table VI-8 Examples of the Characters of the Categories

The Kanji-characters in the input string are marked as PG, SC-1, SC-2, IC or OC according to the above classification. Then the procedure sees if there are any impossible sequences of characters like PC-SC-1, PC-SC-2 and so on, and checks if there is an OC sequence whose length is odd. If such sequences are found, then the procedure decides which attached marks should be changed to more reasonable ones according to the frequencies in the PST.

We applied the procedure to the data which consists of 74,127 Kanji-strings. The results are compared with the manually segmented results. The success rates are shown in Table VI-9.

Length of Compounds	3	4	5	6	7	8	9≥
Success Rate (%)	82.6	87.2	80.0	81.2	66.8	63.4	54.0

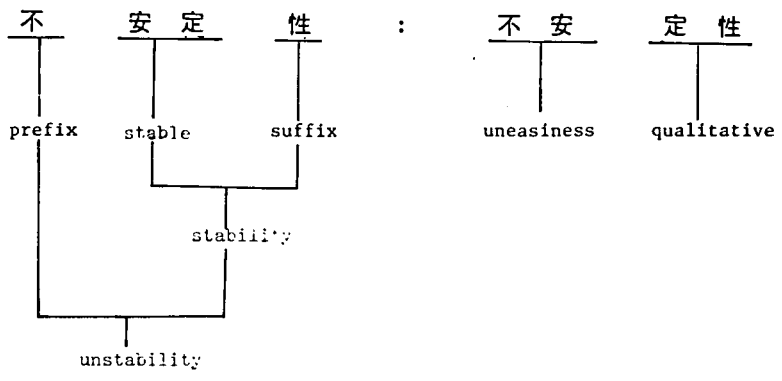
Table VI-9 Results of Segmentation of Compounds (Procedure without Dictionary Consultation)

VI-4-3 Segmentation Procedure of Long Compounds (2) Procedure with Dictionary Consultation

The segmentation procedure (1) gives rather poor results, especially for long compounds. Therefore, we revised the procedure by augmenting it with dictionary consultation. Given a string of Kanji-characters, the revised procedure first looks for any portions of the string to be matched with lexical entries by the dictionary look-up. The procedure tries to segment the string using this result. However, the segmentation at this stage may be ambiguous, because

1. many terms in scientific papers, especially technical terms, are not listed as entries in the dictionary, and
2. a certain sequence of Kanji-characters may have more than one interpretation by lexical entries (see Fig. VI-16).

(1) Input String :



(2) Input String :

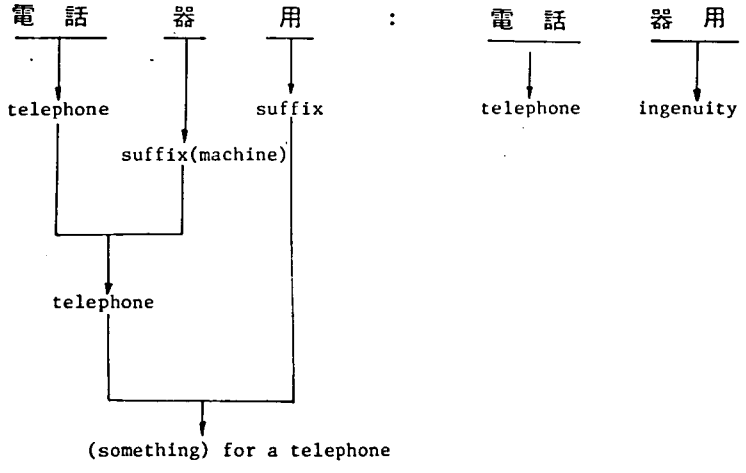


Fig. VI-16 Ambiguous Kanji-strings

If the segmentation result is not unique, the part of the strings which cannot be uniquely segmented will be analyzed further by a certain set of rules which utilize the PST. The flowchart of the procedure and the rules are shown in Fig. VI-17, and Fig. VI-18, respectively.

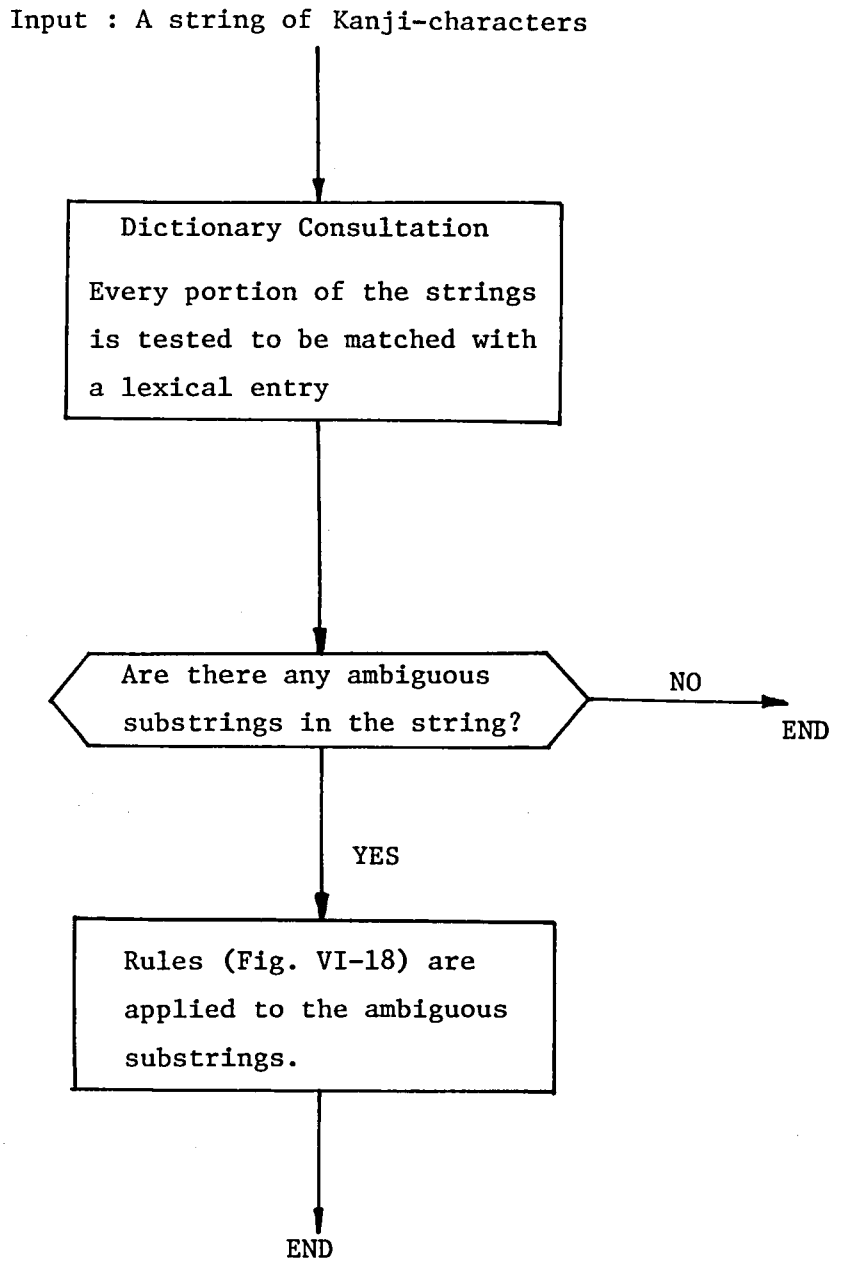


Fig. VI Flow Chart of the Procedure

A, B, C, D, E : Kanji-Characters

a, b, c, d, e : The Value (%) of the Suffix Table for the Kanji-Character

a', b', c', d', e': The Value (%) of the Prefix Table for the Kanji-Character

(1) Rule for a String Composed of Three Kanji-Characters

Input String : .../ ABC / ...

If max(a, a') > max(c, c'),

then /A/BC/ ⟶ the character A is supposed to be a suffix or a prefix character.

otherwise /AB/C/ ⟶ the character C is supposed to be a suffix or a prefix character.

(2) Rule for a String Composed of Four Kanji-Characters

Input String : .../ ABCD / ...

If min(max(a, a'), max(d, d')) > max(b, b', c, c'),

then /AB/CD/ ⟶ the characters A, B, C and D are supposed to be ordinary characters.

otherwise /A/BC/D/ ⟶ the characters A and D are supposed to be suffixes or prefixes.

(3) Rule for a String Composed of Five Kanji-Characters

Input String : ... /ABCDE/ ...

If max(a, a') ≥ max(c, c', e, e'),

then /A/BC/DE/ ⟶ the character A is supposed to be a prefix or a suffix.

otherwise If max(c, c') ≥ max(a, a', e, e'),

then /AB/C/DE/ ⟶ the character C is supposed to be a prefix or a suffix.

otherwise /AB/CD/E/ ⟶ the character E is supposed to be a prefix or a suffix.

Fig. VI-18 Rules for Segmentation

We applied this revised procedure to the same data that the precedure (1) was applied. The score of the results is given in Table VI-10. From this table, we can see that the new procedure works far better than former one. The dictionary which we have used is by no means adjusted for scientific texts. It contains only the words which appear in the ordinary texts, but hardly appear in scientific literatures. Examples of the failures which

are caused by this reason are shown in Fig. VI-19. Examples of the failures caused by the reason that even a few popular technical terms are missing in the dictionary are also given in Fig. VI-20.

Length of Compounds	3	4	5	6	7	8	9≥
Success Rate (%)	94.1	93.6	92.7	91.5	90.0	84.9	78.0

Table VI-10 Results of Segmentation of Compounds (Procedure with Dictionary Consultation)

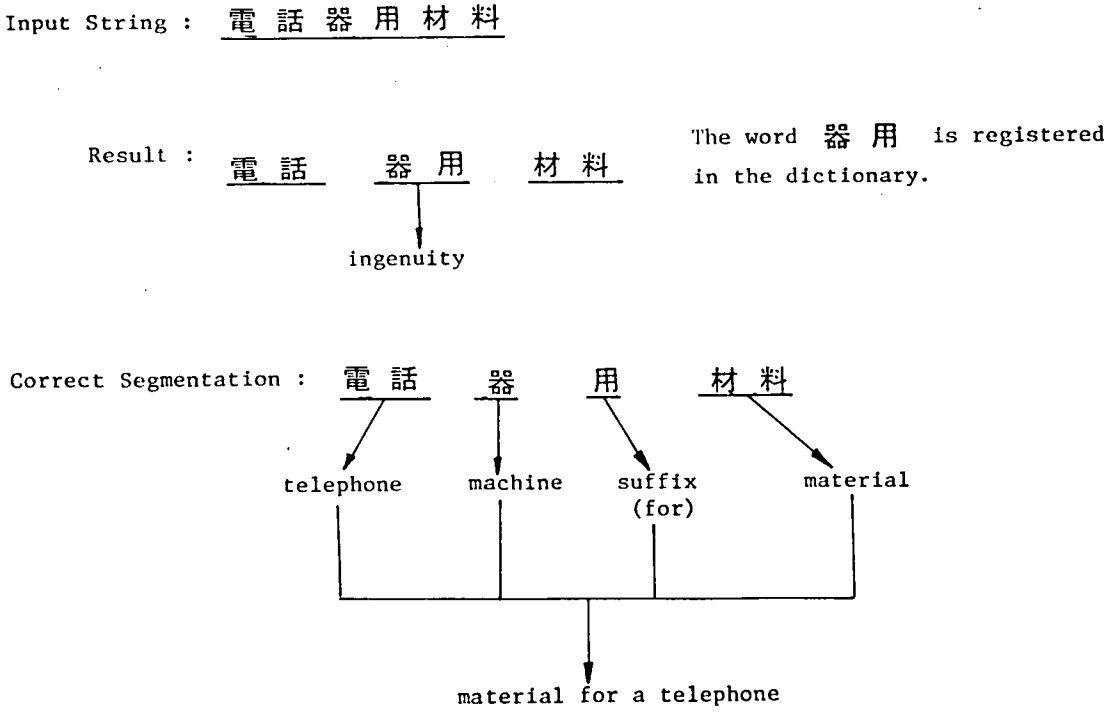


Fig. VI-19 Example of Erroneous Segmentations

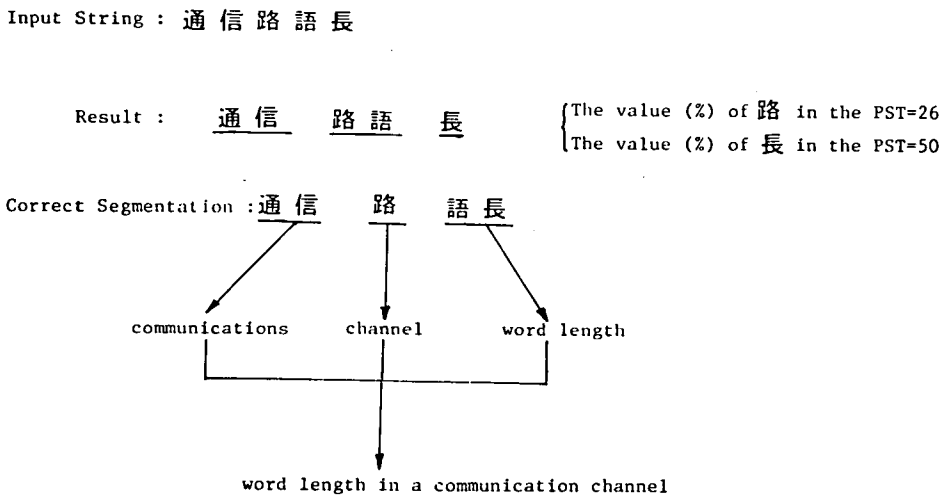


Fig. VI-20 Example of Erroneous Segmentations

If we could use a dictionary of scientific terms, the score would be greatly improved. We believe that this revised procedure is practically usable, if an appropriate dictionary is available, as a preprocessing component of some application systems in language processing, such as automatic translation of scientific papers, automatic indexing and so on.

CHAPTER VII

CONCLUSION

VII-1 Summary of the Thesis

In this thesis, various problems which we encountered during the development of a question-answering system have been discussed. A question-answering system is a typical integrated system. Therefore, aiming at the construction of a question-answering system with natural language input, we have developed several subcomponents of the system such as an analysis program of Japanese sentences (in Chapter III and IV), notations of a semantic network as the internal knowledge representation framework (in Chapter V), a problem solving program using the network (in Chapter V) and a morphological analysis program (in Chapter VI). The primary focus of this thesis has been on the development of an analysis program which utilizes various semantic and contextual information during the analysis process. For the systematic development of the analysis program, we have first developed a new programming language for natural language analysis. The programming language PLATON has several convenient facilities for this purpose (in Chapter II).

Principal results obtained in this research are summarized as follows :

Programming Language PLATON

1. PLATON was developed mainly based on the model of ATN proposed by W. Woods (1970). The language has the attractive feature that the grammatical rules and the control mechanism are clearly discriminated. This enables us to evolve the grammar simply by adding new rules and

to develop the model by adding facilities to its control mechanism.

2. Pattern-matching facility in PLATON makes it easy to write rewriting rules. Moreover, it extracts substructures from the inputs and invokes appropriate semantic and contextual checking functions. The flexible interactions between syntactic and semantic analyses or syntactic and contextual analyses can be easily obtained.

3. PLATON is provided with a flexible backtracking facility. A backtracking mechanism is, we think, necessary for language understanding as in other fields of artificial intelligence. In PLATON, we can easily set up arbitrary numbers of decision points in the program. Then, if subsequent processing results in some failure, control will come back to the points relevant to the cause of the failure.

Analysis Program of Japanese Sentences

4. Because the grammar in the analysis program was written in PLATON, we could easily evolve the grammar through experiments.

5. The lexical descriptions of verbs adopted in the program were based on Case Grammar. Each verb may have one or several activity patterns which actually correspond to case frames of the verb. The descriptions of nouns were based on almost the same notion as the case frames of verbs. Nouns were classified into several categories such as entity nouns, attribute nouns, value nouns, prepositional nouns and action nouns. Different forms of lexical descriptions were devised for different categories.

6. It has been demonstrated that a simple sentence and a long noun phrase in Japanese could be well analyzed through the utilization of the lexical descriptions. Because a verb may have several possible activity patterns, a sentence could instantiate more than one activity pattern. We devised a heuristic function for the selection of the most feasible interpretation of a sentence. There are no sytactic

clues in long noun phrases in Japanese to analyze the relationships among the nouns in the phrases. Therefore, it has been considered for a long time that it was impossible to analyze such long noun phrases by computer. We have demonstrated in this thesis it is possible by utilizing the detailed description of meanings of constituent nouns.

7. We have proposed a semantic network as a representation of contextual information. Events are described in the network by deep case structures (DCS's). We have also devised a structure similar to a DCS for the descriptions of concepts in the network. The network contains various linguistic clues such as case relationships and so on, though the semantic network for problem solving which has been discussed in Chapter V does not contain.

8. The semantic network for contextual information are provided with three kinds of stacks, that is, the noun stack (NS), the hypothetical noun stack (HNS), and the trapping list (TL). We have shown that a certain types of anaphoric expressions and omissions of words or phrases can be treated properly.

Semantic Network as Internal Knowledge Representation

9. The semantic network for the contextual analysis has been organized in the form which is convenient for the utilization in the analysis of sentences. So the descriptions are rich for semantic or contextual analysis of sentences, but sometimes lack necessary information for doing deductions and problem sloving. We have devised a representational framework for such purposes. The representational framework is called S.N.. The S.N. has several advantages over the other frameworks. Various kinds of knowledge, such as algorithmic knowledge, knowledge about external data bases and so on can be naturally embedded in the S.N..

10. The description language for defining the networks has been

given. The description language can be seen as a programming language for problem solving and knowledge representation in general. This programming language is provided with a powerful internal index structure for invoking relevant knowledge in the form of the semantic network S.N..

11. Generalization hierarchy of concepts is a commonly used technique for clustering relevant knowledge. However, the notation of hierarchy developed so far was incomplete in the sense that various complex relationships among concepts cannot be expressed. Aiming at extending the expressive power of generalization hierarchy, we have introduced in the S.N. two new types of nodes, SELF and DISJOINT nodes. These nodes have important role in the process of identifying objects with concepts.

12. The validity of both representation and operations on the networks has been discussed by the corresponding logical formulas.

Morphological Analysis of Japanese Written Texts

13. A dictionary is an important tool for language processing. However, because of the complex writing system of Japanese, there have been no trials until recently to input a big Japanese dictionary in a computer. We have computerized the dictionary 'Shinmeikai Kokugo Jiten' which contains about 70,000 lexical entries, and developed an efficient storage structure for it. In order to utilize the dictionary during morphological analysis, we have provided flexible access methods for the dictionary. We can retrieve a lexical entry by specifying the Kana-, the Kanji- or the mixed spellings of a word. Various variants of inflectional suffixes have been also treated by the dictionary system.

14. Morphological analysis procedure is usually dependent on the specific characteristics of the writing system of a language. The morphological analysis of English mainly concerns about the processing of inflectional variants. Japanese does not have any definite word

separators such as spaces in English. However, Japanese has two different types of characters (Kana and Kanji), and the changes of character types give us a useful clue to segment a long string of characters into smaller and manageable units. We have developed a morphological analysis procedure based on this conception. It has also been demonstrated that the procedure works well on the real texts.

15. Because compounding in Japanese is very frequently used for the expression of complex notions, it is unrealistic to list up all compounds in the dictionary. A morphological analysis procedure of Japanese, therefore, must be provided with the ability of separating a compound word into the constituent lexical words. We have measured the ratios of how often a character is used in real texts as a suffix or a prefix. The segmentation procedure based on the measurement has been developed and the procedure has obtained good results for real texts.

VII-2 Areas for Future Work

We have summarized above the research and experimental results obtained in our study. We hope that this work will give a hopeful future to language understanding systems. However, we admit that there still remain a lot of difficult problems to be solved before computer systems become to be able to communicate in fluent natural language with human users. At the same time, we also admit that there are many improvements which can be made within the framework of the current research. Possible improvements are listed below :

1. Introduction of the abilities of logical deductions or problem solving to the analysis of sentences. The present analysis program does not utilize the abilities of logical deductions or problem

solving based on the S.N.. The introduction of such abilities to the analysis program is expected to be useful for resolving certain types of ambiguities.

2. Development of a framework for the representation of more global knowledge. Such framework should be needed for both the analysis program and the problem solving program. For the analysis program, we need a framework which corresponds to Scrip, or Frame by Schank or others. For the problem solving program, we need a framework in which we can define strategies.

3. Augmentation of the description language for the S.N.. There are several deficiencies in the current version of the S.N.. Especially, we must provide a much more powerful and descriptive framework for defining new predicates in the S.N..

4. Development of more flexible control mechanisms for problem solving. The current version of problem solving program is constructed basically on the notion of the backward reasoning and the conventional AND-OR hierarchies of problems. We should provide more flexible and powerful mechanisms such as parallel computation, mixed mode of backward and forward reasonings, and so on.

5. Introduction of semantic information to the analysis of a long compound in Japanese. The results of the segmentation of long compounds are relatively good. However, in order to determine the relationship among the constituent words of a compound, we must provide each lexical entris in the dictionary with a certain description of the meaning in a computer usable form.

Although there will be many difficult problems in the development of the model of language understanding, we hope in conclusion that our research will be helpful in developing more powerful and comprehensive language understanding systems in future.

REFERENCES

[Bobrow 1974] D.G.Bobrow, "New Programming Languages for Artificial Intelligence Research," *Computing Survey*, Vol. 16, No. 3, 1974

[Bruce 1975] B.Bruce, "Case System for Natural Language," *Jour. of Artificial Intelligence*, Vol. 6, 1975

[Celce 1972] M.Celce, "Paradigm for Sentence Recognition," *SDC Report HRT-15092/7907*, 1972

[Chang 1973] C.L.Chang and R.C.Lee, "Symbolic Logic and Mechanical Theorem Proving," *Academic Press*, 1973

[Charniak 1972] E.Charniak, "Towards a Model of Children's Story Comprehension," *ph.D Thesis, MIT*, 1972

[Charniak 1977] E.Charniak, "Ms. Maloprop, A Language Comprehension Program," *Proc. of 5th IJCAI*, Cambridge, 1977

[Chomsky 1957] N.Chomsky, "Syntactic Structure," *Mouton, The Hague*, 1957

[Chomsky 1965] N.Chomsky, "Aspects of the Theory of Syntax," *MIT Press, Cambridge*, 1965

[Coles 1969] L.S.Coles, "Talking with a Robot in English," *Proc. of 1st IJCAI*, Washington D.C., 1969

[Colmerauer 1971] A.Colmerauer, "Les System-Q ou un Formalisme pour Analyser et Synthetizer des Phrases sur Ordinateur," *Project de Traduction Automatique de l'Montreal*, TAUM 71, 1971

[Fillmore 1968] C.J.Fillmore, "The Case for Case," *in Universals in Linguistic Theory (eds.Bach and Harms)*, North Holland, 1968

[Hendrix 1975a] G.G.Hendrix, "Partitioned Network for the Mathematical Modeling of Natural Language Semantics," *ph.D Thesis, University of Texas*, 1975

[Hendrix 1975b] G.G.Hendrix, "Expanding the Utility of Semantic Networks through Partitioning," *Proc. of 4th IJCAI*, Tbilisi, 1975

[Hendrix 1977] G.G.Hendrix and R.Fike, "A Network-Based Knowledge Representation and Its Natural Deduction System," *Proc. of 5th IJCAI*, Cambridge, 1977

[Kuno 1963] S.Kuno and A.G.Oettinger, "Syntactic Structure and Ambiguity of English," *Proc. of AFIPS 1963 FJCC*, Vol. 24, 1963

[Kuno 1966] S.Kuno, "The Augmented Predictive Analyzer for Context-Free Languages - Its Relative Efficiency," *C.ACM*, Vol. 9, No. 11, 1966

[Minker 1977] J.Minker and J.McSkimin, "The Use of Semantic Network in a Deductive Question-Answering System," *Proc. of 5th IJCAI*,

Cambridge, 1977

[Minsky 1975] M.Minsky, "A Framework for Representing Knowledge," *AI Memo 306, MIT,* 1974

[Mylopoulos 1975] J.Mylopoulos, "TORUS - A Natural Language Understanding System for Data Management System," *Proc. of 4th IJCAI,* Tbilisi, 1975

[Mylopoulos 1977] J.Mylopoulos, "An Overview of a Procedural Approach to Semantic Networks," *Proc. of 5th IJCAI,* Cambridge, 1977

[Norman 1973] D.Norman and D.E.Rumelhart, "Active Semantic Network as a Model of Human Memory," *Proc. of 3rd IJCAI,* Stanford, 1973

[Norman 1975] D.Norman and D.E.Rumelhart, "Exploration in Cognition," *Freeman,* 1975

[Pratt 1973] V.R.Pratt, "Linguistic Oriented Programming Language," *Proc. of 3rd IJCAI,* Stanford, 1973

[Pratt 1975] V.R.Pratt, "LINGOL - A Progress Report," *Proc. of 4th IJCAI,* Tbilisi, 1975

[Quillian 1968] R.Quillian, "Semantic Memory," *in Semantic Information Processing(ed. Minsky),* MIT Press, 1968

[Quillian 1969] R.Quillian, "The Teachable Language Comprehender," *C.ACM,* Vol. 12, No. 8, 1969

[Raphael 1968] B.Raphael, "A Computer Program for Semantic Information Retrieval," *in Semantic Information Processings(ed. Minsky),* MIT Press, 1968

[Sandewall 1970] E.Sandewall, "A Set-Oriented Property Structure Representation for Binary Relations," *Machine Intelligence 5,* American Elsevier, 1970

[Schank 1973] R.C.Schank, N.Goldman, C.Rieger, and C.Riesbeck, "MARGIE : Memory, Analysis, Response, Generation, and Inference on English," *Proc. of 3rd IJCAI,* Stanford, 1973

[Schank 1975a] R.C.Schank, R.P.Abelson, "Scripts, Plans and Knowledge," *Proc. of 4th IJCAI,* Tbilisi, 1975

[Schank 1975b] R.C.Schank, "SAM - A Story Understander," *Research Report 43, Yale Univ.,* 1975

[Schank 1975c] R.C.Schank, "Conceptual Information Processing," North Holland, 1975

[Schubert 1975] L.K.Schubert, "Toward a State Based Conceptual Representation," *Proc. of 4th IJCAI,* Tbilisi, 1975

[Schubert 1976] L.K.Schubert, "Extending the Expressive Power of Semantic Networks," *Artificial Intelligence,* Vol. 7, 1976

[Scragg 1976] G.Scragg, "Semantic Nets as Memory Model," *in Computational Semantics (eds. Charniak and Wilks)*, North Holland, 1976

[Simmons 1973] R.F.Simmons, "Semantic Network : Their Computation and Use for Understanding English Sentences," *in Computer Models of Thought and Language (eds. Schank and Colby)*, Freeman, 1973

[Simmons 1975] R.F.Simmons *et. al.*, "Semantically Analyzing an English Subset for the Clowns Micro World," *American Jour. of Computational Linguistics* Microfiche 18, 1975

[Simmons 1977] R.F.Simmons and D.Chester, "Inferences in Quantified Semantic Networks," *Proc. of 5th IJCAI*, Cambridge, 1977

[Sowa 1976] J.Sowa, "Conceptual Graphs for a Data Base Interface," *IBM Jour. of Research and Development*, Jul. 1976

[Tanaka 1977] H.Tanaka, T.Sato, F.Motoyoshi, "A Programming System for Natural Language Processing - on Extended LINGOL," *Trans. IECE Japan*, Vol. J60-D, No. 12, 1977

[Weizenbaum 1966] J.Weizenbaum, "ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine," *C.ACM*, Vol. 9, No. 1, 1966

[Wilks 1975a] Y.Wilks, "An Intelligent Analyzer and Understander of English," *C.ACM*, Vol. 18, 1975

[Wilks 1975b] Y.Wilks, "A Preferential, Pattern-Matching Semantics for Natural Language Inference," *Artificial Intelligence*, Vol. 6, 1975

[Wilks 1976] Y.Wilks, "Frames, Scripts and Fantasies," *Proc. of 6th ICCL*, Ottawa, 1976

[Wilks 1977] Y.Wilks, "Knowledge Structure and Language Boundaries," *Proc. of 5th IJCAI*, Cambridge, 1977

[Winograd 1972] T.Winograd, "Understanding Natural Language," *Academic Press*, 1972

[Winograd 1975] T.Winograd, "Frame Representation and the Declarative/Procedural Contraversy," *in Representation and Understanding (eds. Bobrow and Collins)*, Academic Press, 1975

[Woods 1970] W.A.Woods, "Transition Network Grammar for Natural Language Analysis," *C.ACM*, Vol. 13, 1970

[Woods 1972] W.A.Woods, "The Lunar Sciences Natural Language Information System," *BBN Report No. 2378*, 1972

[Woods 1975] W.A.Woods, "What's in a Link," *in Representation and Understanding (eds. Bobrow and Collins)*, Academic Press, 1975

PUBLICATIONS AND TECHNICAL REPORTS BY THE AUTHOR

PUBLICATIONS

(1) M.Nagao and J.Tsujii, "Mechanism of Deduction in a Question-Answering System with Natural Language Input," *Proc. of 3rd IJCAI*, Stanford, Aug. 1973(in English)

(2) M.Nagao and J.Tsujii, "A New Programming Language for Natural Language Analysis," *Jour. of IPS Japan*, Vol. 15, No. 9, 1974 (in Japanese)

(3) M.Nagao and J.Tsujii, "PLATON - A New Programming Language for Natural Language Analysis," *Proc. of 2nd USA-JAPAN Computer Conference*, Tokyo, Aug. 1975(in English)

(4) M.Nagao and J.Tsujii, "Semantic Analysis of Japanese Sentences," *Jour. of IPS Japan*, Vol. 17, No. 1, 1976(in Japanese)

(5) M.Nagao, J.Tsujii and K.Tanaka, "Contextual Analysis of Japanese Sentences," *Jour. of IPS Japan*, Vol. 17, No. 1, 1976(in Japanese)

(6) M.Nagao and J.Tsujii, "Analysis of Japanese Sentences by Using Semantic and Contextual Information," *American Journal of Computational Linguistics*, Microfiche 41 , Jan. 1976(in English)

(7) M.Nagao and J.Tsujii, "PLATON - A New Programming Language for Natural Language Analysis," *American Journal of Computational Linguistics*, Microfiche 37, Jan. 1976(in English)

(8) M.Nagao and J.Tsujii, "Analysis of Japanese Sentences by Using Semantic and Contextual Information," *Proc. of 6th ICCL*, Ottawa, Jul. 1976(in English)

(9) J.Tsujii, "Problem Solving Techniques in Robotics," *Special Issue for Robot, Bit*, Jul. 1976(in Japanese)

(10) J.Tsujii, "Computers and Language," *Mathematical Sciences*, No. 168, Jun. 1977(in Japanese)

(11) M.Nagao and J.Tsujii, "Survey of Current Computational Linguistics and Natural Language Understanding Researches," *Jour. of IPS Japan*, Vol. 18, No. 1, 1977(in Japanese)

(12) M.Nagao, J.Tsujii, A.Yamagami and S.Tatebe, "Dictionary Organization and Morphological Analysis," *Jour. of IPS Japan*(in Japanese, to appear)

TECHNICAL REPORTS

(13) T.Sakai, M.Nagao and J.Tsujii, "Information Network for Binary Relations and a Question-Answering System Using it," *National Convention Record of IECE Japan*, May 1972

(14) M.Nagao and J.Tsujii, "Deductive Question-Answering System," *Technical Report of the Professional Group on Automata and Language of IECE Japan*, AL72-118, Jan. 1973

(15) M.Nagao and J.Tsujii, "A Deductive Question-Answering System," *National Convention Record of IECE Japan*, May 1973

(16) M.Nagao and J.Tsujii, "A Programming Language for Natural Language Analysis and its Application for Japanese Noun Phrase Analysis," *Technical Report of the Professional Group on Automata and Language of IECE Japan*, AL73-71, Jan. 1974

(17) M.Nagao, J.Tsujii and K.Nakamura, "Analysis of Japanese Sentences Based on Case Grammar," *Technical Report of the Professional Group on Computational Linguistics of IPS Japan*, CL4-1, May 1974

(18) M.Nagao, J.Tsujii and K.Tanaka, "Contextual Analysis for a Noun Phrase," *National Convention Record of IPS Japan*, Dec. 1974

(19) M.Nagao and J.Tsujii, "Natural Language Analysis and Question-Answering Systems," 1975 *Joint Convention Record of Four Institutes of Electrical Engineers of Japan*, Sept. 1975

(20) M.Nagao, J.Tsujii and K.Tanaka, "Analysis of Input Sentences in a Question-Answering System," *National Convention Record of IPS Japan*, Nov. 1975

(21) M.Nagao, J.Tsujii and K.Tanaka, "Question-Answering System and the Analysis of the Input Sentences," *Technical Report of the Professional Group on Automata and Language of IECE Japan*, AL75-51, Nov. 1975

(22) M.Nagao and J.Tsujii, "Survey of Current Computational Linguistics," *Technical Report of the Professional Group on Computational Linguistics of IPS Japan*, CL7-1, Sept. 1976

(23) M.Nagao, J.Tsujii and A.Terada, "Internal Data Representation of a Question-Answering System," *Technical Report of the Professional Group on Automata and Language of IECE Japan*, AL76-45, Oct. 1976

(24) M.Nagao, J.Tsujii, A.Yamagami and S.Tatebe, "Support System for Natural Language Processing II," *Technical Report of the Professional Group on Automata and Language of IECE Japan*, AL77-25, Jul. 1977

(25) M.Nagao, J.Tsujii, A.Yamagami and S.Tatebe, "Dictionary Organization for Morphological Analysis of Japanese Written Texts," *National Convention Record of IECE Japan*, Aug. 1977

(26) M.Nagao, J.Tsujii, A.Yamagami, S.Tatebe and M.Hoda, "Analysis of Japanese Long Compounds," *National Convention Record of IECE Japan*, Aug. 1977

(27) M.Nagao, J.Tsujii, A.Yamagami, S.Tatebe and M.Hoda, "Morpho-

logical Analysis of Japanese Written Texts," *National Convention Record of IECE Japan*, Aug. 1977

(28) M.Nagao and J.Tsujii, "Algorithms for Natural Language Understanding," 1977*Joint Convention Record of Four Institutes of Electrical Engineers of Japan*, Oct. 1977

(29) M.Nagao, J.Tsujii and A.Terada, "Expressive Power of a Semantic Network and Problem Solving Techniques by Using it," *Technical Report of the Professional Group on Automata and Language of IECE Japan*, AL77-43, Oct. 1977

(30) M.Nagao, J.Tsujii and A.Terada, "Deduction Process Based on a Semantic Network," *National Convention Record of IPS Japan*, Nov. 1977