

LOGIC BASED JAPANESE SENTENCE
ANALYSIS

RYOICHI SUGIMURA

1997

②

LOGIC BASED JAPANESE SENTENCE ANALYSIS

RYOICHI SUGIMURA

1997

ABSTRACT

Natural language is crucial for all human activities, especially those in intellectual realms. This fact means that research into natural language processing has too many aspects for us to gain an overall perspective. A simple glance will easily show that this area has been enriched by researchers not only from the field of linguistics but also philosophy, psychology, and technology. Therefore we need to set some strong hypothetical principles to navigate our way through this interdisciplinary research area.

It is well known that to observe some natural phenomena, we have to approach the phenomena from some viewpoint. However, this viewpoint constrains the whole scale of understanding of the phenomena. Some features of that viewpoint may distort the true nature of the target object. This is a dilemma between a researcher and their object. Therefore, when we select a viewpoint we should keep it in mind that while we will learn something new, we may simultaneously be disregarding or distorting other features. However, we have to hope that the accumulation of results from a variety of research viewpoints will allow us to gain wider knowledge and new frontiers.

In this thesis, the main research vehicle is logic programming. Logic has a long history from ancient Greece, and plenty of research results have been accumulated in Western history. Together with Christianity, logic can be regarded as one of the most important pillars of western culture. Unfortunately, up to the age of modern civilization, logic tended to be regarded as a device for deception, used by eloquent people. However, after the 19th century, there emerged a vast amount of prominent research results from logic. For example, modern mathematics has its base in logic, and logical contradiction sometimes played a very important role in promoting the acquisition of new knowledge. Therefore, logic as a research compass or a probe is not a protruding nail in the research main stream, but a fairly normal vehicle.

Logic programming has its base in logic, more specifically Horn clause logic. Therefore, if we utilize logic programming, it will become slightly easier for us to constrain our research activities to the scope of logic. Moreover logic programming has the benefit that logic programming languages have a

promising future in regard to parallel processing. Therefore, research activities on logic programming afford us a strong compass and will hopefully allow us to break through a technological frontier and realize *parallel computing*.

The works presented in this thesis have their base in my work at ICOT (Institute for New Generation Computer Technology), and at Matsushita Electric Industrial Co., Ltd.

Armed with logic programming, I decided to tackle natural language analysis at ICOT. At that time, it seemed very hard for me to develop all the tools needed for natural language analysis. For that reason, I proposed a new project named "Language Tool Box", and I asked my colleagues in ICOT to join this project. Fortunately, I received much support for this project from researchers, and I worked as a researcher who was responsible for Japanese sentence analysis and as a research leader of this project.

The first thing I tried was to make new fundamental tools for natural language analysis. At beginning of the research, I decided to develop a dictionary, morphological analyzer, and a parser using logic programming language. This is not because I had some theoretical reason to support such development but because they seemed inevitable to the research. I would like to avoid research activities without roots deeply stuck in actual natural language. *The main aim of my research was to attain a new algorithm which could be implemented both in sequential and parallel logic programming language.* As a research plan to launch these three tools, I decided to investigate a morphological analyzer because this part is strongly related to the dictionary part and parser. Consequently, my research activities were focused on the morphological analyzer first at ICOT.

In this thesis, a morphological analyzer will be presented first.

Together with development of the fundamental tools, I was trying to resolve diversified research directions by way of logic. There was a tremendous amount of distinguished research results based on many academic policies. First of all, my research colleague and I in ICOT decided to focus on discourse processing that might give us a new view of natural language processing. Next, we started to study a variety of issues of natural language analysis using logical devices. One of the most important issues for us was disambiguation, which had posed a complicated riddle for researchers in the

field of sentence analysis. I tried to grasp previous available knowledge about Japanese sentence analysis by new tools based on logic. At that time, the unification grammar formalism emerged on the West Coast of the United States, and many researchers paid keen interest toward this new framework. There was no reason to neglect this new framework for me, but I personally prefer the traditional framework based on "kakari-uke" as dependency relation. This research direction came from my intuition which did not allow me to hypothetically consider the existence of grammatical categories like verb phrase or noun phrase or higher grammatical categories.

Therefore, I decided to concentrate my study on dependency relations. First of all, I tried to represent dependency relations in the framework of logic programming. During this approach, I encountered a constraint programming paradigm which allowed the treatment of dependency relations in a new way by which the solution of ambiguities could be postponed until a later time when we could get more appropriate information. At that time, no one knew how to express the crucial relations of dependency analysis in the form of logic. After some effort, I found a way to express dependency relations exactly in the form of logical formula. As a result, I could express crucial issues of sentence analysis in logic, and it became easier for us to grasp what are the central matters which make discourse processing difficult.

In the second chapter, I will outline the research on Japanese sentence analysis.

All the results of these research activities were collected into an experimental system named "DUALS" which stands for "Discourse Understanding Aimed at Logic based Systems". In an attempt to apply new technologies toward issues of natural language processing, we found many actual problems and got a view toward the next step. One of the results of DUALS was realization of the importance of making new machinery which handles in a unified way the several kinds of information which are usually scattered into tiny parts of the system. Another result was the need to expand the logical framework. These matters will be discussed in the third chapter of DUALS.

After the completion of DUALS-III, I came back to Matsushita and took on the development of a project to develop machine translation systems as a project leader and a researcher responsible for Japanese sentence analysis.

As a general research principle I believe research ideas can be genuine research results when they are applied to the actual product.

With the ongoing research activities at ICOT mentioned above, my colleagues at Matsushita and I tried to apply some of the results of research at ICOT to the machine translation system. First of all, we tried to grasp issues which make machine translation hard to carry out. These efforts were unified to develop a machine translation system with a dialogue interface which assists the machine by employing human knowledge. We rearranged and checked what are the most intricate parts for the machine to handle properly. We applied our new ideas to the list of these issues one by one. First of all, we applied the morphological analyzer to our machine translation system and succeeded in achieving a high performance in analysis.

To summarize the story of this thesis, a morphological analyzer named "LAX" will be introduced first. Then, a new way to handle sentence ambiguities will be shown and discussed. After the representation of two frameworks, DUALS will be represented with the focus on sentence analysis. Before the concluding remarks, the machine translation system will be presented and discussed.

Acknowledgments

I would like to express my sincere appreciation to Professor Makoto Nagao of Kyoto University for his supervision and encouragement to complete this thesis.

I also would like to thank professor Yuji Matsumoto of Advanced Institute of Science and Technology, Nara, for continuous encouragement and useful discussions when we worked together in ICOT, and after I was back in Matsushita.

I am grateful to Professor Hozumi Tanaka of Tokyo Institute of Science and Technology for his giving me help to promote my research.

I am grateful to Professor Shinji Tomita for his warm advice and assistance in completing this thesis.

I am also grateful to Mr. Miki, and Mr. Sakao for giving me ample time to finish this thesis.

I am also grateful to Mr. Cornish for giving me fruitful suggestion concerning my research documents and research ideas as my intimate research colleague.

Contents

ABSTRACT

ACKNOWLEDGMENTS

CONTENTS

1. MORPHOLOGICAL ANALYZER LAX	1
1.1. INTRODUCTION	1
1.2. FUNDAMENTAL ALGORITHM	2
1.3. RULES AND DICTIONARY	5
1.3.1. <i>Rules for Concatenation</i>	6
1.3.2. <i>Expansion of NFSA</i>	11
1.3.3. <i>Format of the Dictionary</i>	12
1.4. THE ELEMENTARY IMPLEMENTATION	13
1.5. EFFECTIVE IMPLEMENTATION	16
1.5.1. <i>Making Processes Activation Demand-Driven</i>	17
1.5.2. <i>Data and Its Flow in the Practical Algorithm</i>	20
1.5.3. <i>Complexity</i>	26
1.6. HEURISTICS FOR DISAMBIGUATION	29
1.7. IMPLEMENTATION ON LOGIC PROGRAMMING LANGUAGE	32
1.7.1. <i>Implementation on Prolog</i>	32
1.7.2. <i>Implementation on PSI</i>	33
1.8. IMPLEMENTATION ON C	35
1.8.1. <i>Implementation on a Workstation</i>	35
1.8.2. <i>Implementation on a PC</i>	37
1.9. EVALUATION	38
1.9.1. <i>Quantitative and Qualitative Evaluation</i>	38
1.9.2. <i>Discussion</i>	41
2. JAPANESE DEPENDENCY ANALYSIS	43
2.1. INTRODUCTION	43
2.2. TOOLS FOR JAPANESE SENTENCE ANALYSIS	44
2.2.1. <i>Solver for Propositional Logic</i>	44
2.2.2. <i>Logic Based Parser SAX</i>	47
2.3. BASIC PRINCIPLES OF DEPENDENCY ANALYSIS	48
2.4. DEPENDENCY ANALYSIS USING PROPOSITIONAL LOGIC	49
2.4.1. <i>Basic Algorithm</i>	50
2.4.2. <i>Implementation</i>	51
2.4.3. <i>Analysis by Syntactic and Semantic Constraints</i>	52

2.4.4. Contextual Analysis of a Sentence by Logical Constraints.....	56
2.5. DEPENDENCY ANALYSIS USING TYPE 0 RULES.....	59
2.5.1. Basic Algorithm.....	59
2.5.2. Grammar Rules for Dependency Structure.....	60
2.5.3. Implementation.....	61
2.6. CONSTRAINT ANALYSIS OF LOGIC GRAMMARS.....	63
2.6.1. Logic Grammar.....	64
2.6.2. Multiple Interpretations and Context.....	65
2.6.3. Constraints on Discourse.....	66
2.6.4. Sentence Analysis.....	66
2.6.5. Constraint Analysis in Discourse.....	67
2.6.6. Phrase Structure Analysis in Discourse.....	67
2.6.7. Dependency Structure Analysis in Discourse.....	70
2.7. RELATED METHODS.....	71
2.7.1. Dependency Analysis Using Modification Deciding Network.....	71
2.8. COMPARISON, EVALUATION AND DISCUSSION.....	72
3. JAPANESE SENTENCE ANALYSIS IN EXPERIMENTAL DISCOURSE UNDERSTANDING SYSTEM DUALS-III.....	75
3.1. INTRODUCTION.....	75
3.2. LANGUAGE TOOL BOX LTB.....	76
3.2.1. Introduction.....	76
3.2.2. Structure of LTB.....	77
3.2.3. LTB Shell.....	78
3.3. CONFIGURATION OF DUALS-III.....	79
3.4. LINGUISTIC KNOWLEDGE BASE.....	81
3.4.1. Overview of the Language Knowledge Base.....	81
3.4.2. Master Dictionary.....	82
3.4.3. Future Research for the Linguistic Knowledge Base:.....	83
3.5. SEMANTIC DESCRIPTION.....	84
3.5.1. Overview of CIL in DUALSCIL.....	84
3.5.2. Partially Specified Terms and Their Operation.....	84
3.5.3. Future Research for the Semantic Description Language.....	86
3.6. MORPHOLOGICAL AND SEMANTIC ANALYSIS.....	86
3.6.1. Overview of the Morphological Analyzing System (LAX).....	86
3.6.2. Description Format of the Morphological and Semantic Knowledge.....	87
3.6.3. Special Features of Morphological and Semantic Knowledge.....	87
3.6.4. Future Research on Morphological Analysis.....	89
3.7. SYNTACTIC AND SEMANTIC ANALYSIS.....	90
3.7.1. Syntactic Analysis System (SAX).....	90
3.7.2. Description of Syntactic and Semantic Knowledge.....	90
3.7.3. Special Features of Syntactic and Semantic Analysis.....	90
3.7.4. Handling Ambiguity.....	91

3.7.5. <i>Future Research in Syntactic and Semantic Analysis</i>	91
3.8. PROBLEM SOLVING	92
3.8.1. <i>Summary of Problem Solving</i>	92
3.8.2. <i>Description of Knowledge for Problem Solver</i>	92
3.8.3. <i>Special Features of Knowledge for Problem Solver</i>	94
3.8.4. <i>Future Research on Problem Solving</i>	96
3.9. EVALUATION AND DISCUSSION	96
4. JAPANESE SENTENCE ANALYSIS IN JAPANESE ENGLISH MACHINE	
TRANSLATION	99
4.1. INTRODUCTION	99
4.2. USER INTERFACES TO GET HUMAN KNOWLEDGE	100
4.2.1. <i>Morphological Analysis</i>	101
4.2.2. <i>Japanese Sentence Analysis</i>	103
4.3. EVALUATION AND DISCUSSION	106
5. CONCLUSION	109
5.1. DISCUSSION AND FUTURE WORK.....	109
6. LIST OF PUBLICATIONS	113
6.1. LIST OF MAJOR PUBLICATIONS.....	113
6.2. LIST OF OTHER PUBLICATIONS.....	115
6.3. RELATED WORK.....	116
7. APPENDIX A	123
8. APPENDIX B	127
9. APPENDIX C	129

List of Figures

Figure 1: Concatenation of morphemes	7
Figure 2 : An Analysis of a word by a multiple transition table.....	11
Figure 3 : Multi Processes in a well formed sub-string table	14
Figure 4 : An Example of a “dict” processes corresponding to input characters	17
Figure 5 : TRIE Structure for a Lexical Dictioanry	19
Figure 6 : An Implementation of a Dictionary on Prolog.....	19
Figure 7 : “Dict” Processes connected by Data Streams.....	21
Figure 8 : Definition of The Data Stream Structure.....	21
Figure 9 : An Example Data Stream.....	23
Figure 10 : Packed Data	24
Figure 11 : Process Layout for Lexical Analysis.....	26
Figure 12 : An Example of Results of Lexical Analysis.....	30
Figure 13 : LAX System on PSI Machine	34
Figure 14 : An Example of Retrieval of sub-strings from a Dictionary	36
Figure 15 : The Retrieval System using Cache for TRIE Structures	36
Figure 16 : The Evaluation System of the Morphological Analyzer.....	39
Figure 17 : An Example of Illegal Modifying Relations.....	49
Figure 18 : An Example of a Modification Deciding Table	50
Figure 19 : Constraints on a Modification Deciding Table	51
Figure 20 : Resultant Constraints on a Modification Deciding Table.....	52
Figure 21 : An Example of Semantic Constraints	53
Figure 22 : An Example of Semantic Check on Modifying Relation	54
Figure 23 : An Example of Ambiguous Semantic Relations	55
Figure 24 : Examples of Ambiguities in Syntactic Analysis	56

Figure 25 : An Example of Syntactic Constraints between Two Interpretations	58
Figure 26 : An Example of Dependency Structure realized by Rewriting Rules.....	61
Figure 27 : An Example of Dependency Analysis.....	63
Figure 28 : An Example of Ambiguities of Sentence Analysis	68
Figure 29 : An Example of Valid Interpretations in Ambiguous Sentence Analysis	71
Figure 30 : An Example of Procedural Disambiguation	73
Figure 31 : Configuration of the LTB.....	77
Figure 32 : Data Flow Model for LTB Shell.....	79
Figure 33 : Configuration of DUALS and LTB.....	80
Figure 34 : Human Interface of DUALS III	81
Figure 35 : An Example result of morphological and semantic analysis of DUALS-III	88
Figure 36 : An Example of DCG rules and its Parsed Tree	90
Figure 37 : An Example of Semantic Representation of DUALS-III.....	95
Figure 38 : Human Interface for Morphological Analysis of the Machine Translation System	102
Figure 39 : Human Interface for Word Meaning Disambiguation	105
Figure 40 : Human Interface for Disambiguation of Modifying Relations	106

List of Tables

Table 1: An Example of A CYK Table.....	3
Table 2 : An Example of Connection Matrix.....	4
Table 3 : An Example of CYK Table.....	5
Table 4 : An Example of a Lexical Dictioanry	18
Table 5 : An Example of Resultant Data Structure of Lexical Analysis.....	31
Table 6 : Comparison of Three implementations of LAX Algorithm	40
Table 7 : Number of Constraints corresponding to Length of Input Words .	51

1. Morphological Analyzer LAX

1.1. Introduction

In this section, a method of Japanese morphological analysis which runs on a sequential and parallel logic programming language such as Prolog, ESP [Chikayama], or KL1[Ueda] [Sugimura 87c][Sugimura 87d], and sequential procedural language such as C will be presented[Sugimura 88b][Tanaka]. In the technique presented here, a morphological dictionary is transformed into an executable program with a compacted data structure in Prolog, ESP or C. The basic algorithm which is illustrated here has its base in the CYK algorithm[Aho]. In order to realize high performance, it also utilizes the packed data structure [Tomita] which enables the elimination of duplicate handling of the same data structure in the analyzing process.

In the installation in the logic programming language, the dictionary is installed in the form of TRIE data structure [Kamiwaki] in which morphemes can be quickly looked up using a fast index search mechanism which is provided in a logic programming language like Prolog. On the other hand, in the installation in the procedural language, the dictionary is installed using character index and a TRIE retrieval mechanism which enables us to obtain compact size and high retrieval speed.

1.2. Fundamental Algorithm

A parallel algorithm obviously should also be effective in sequential execution from the point of view of space and time complexity, because, even though we could utilize parallel machines, their effects would be proportional to the number of machines at most. Therefore, I will present the basic algorithm comparing it with the well known sequential algorithm 'CYK' in which the algorithm can be explained in a sequential and declarative way.

At first, the fundamental algorithm will be shown by using examples aiming at an overall understanding of the algorithm.

Let us give an input string to be analyzed as [く, る, ま, で, ま, つ] (*ku, ru, ma, de, ma, tu*). The first procedure is to look up the morphological dictionary in order to discover the morphemes which can cover the sub-strings of the input string. In the example, each character of the input morpheme is numbered from the first up to the number which is the length of the input string. If morphemes are found which correspond to the sub-strings, they are registered in the analysis table usually called "CYK table" in the first row as shown in Table 1. For notational convenience, morphemes found will be written in the form of an index number with string and grammatical category like "index(morpheme, Grammatical Category)".

In the example, from the head of the input string, a set of four morphemes {a(車,N), b(包,Vs), c(来,Vs), d(刳,Vs)} which cover the sub-string are found. A noun '車'(*kuruma*) <car> covers from position 1 to 3, therefore it is registered in the first row of the third column. A verb stem '包'(*kuru*) <to wrap> covers from position 1 to 2, therefore it is registered in the first row of the second column. Verb stems '刳'(*ku*) <to hollow out> and '来'(*ku*) <to come> cover from position 1 to 1, therefore they are registered in the first row of the first column.

Table 1: An Example of A CYK Table

6						
5						
4						
3	a(車,N)					
2	b(包,V)		i(まで,P)	k(デマ,N)	n(松,N)	
1	c(来,Vs) d(例,P)	e(流,Vs) f(る,P)	g(間,N) h(巻,Vs)	j(出,Vs) l(で,P)	m(間,N) o(待,Vs)	p(津,N) q(釣,Vs) r(つ,Vs)
	く 1	る 2	ま 3	で 4	ま 5	つ 6

After morpheme dictionary consultation, an acceptable sequence of Japanese morphemes will be derived on the CYK table. The acceptable sequence of Japanese morphemes is defined using a connection matrix[Aizawa] as shown in following example. In the connection matrix, Cartesian products of morphemes are examined as to whether a pair of morphemes can be concatenated to make a sequence of morphemes in Japanese words. In the example, a group of morphological words which can be the left hand side of some morpheme is listed up in the first column, and a group of morphological words which can be the right hand side of some morpheme is listed up in the first row. For example, for the purpose of checking whether c(来,Vs) and f(る,P) can be concatenated or not, look up the matrix position in a column which has c(来,Vs) as its first row, and a row which has f(る,P) as its first column. If there stands a bit which is in the example represented as “○”, a sequence of two words c(来,Vs) and f(る,P) could be an acceptable Japanese string.

Table 2 : An Example of Connection Matrix

	c(来,Vs) d(剥,P) b(包,V) a(車,N)	f(る, P)	i(まで P)	l(で,P)	待	松	つ	end
start	○				○	○		
c(来,Vs)		○						
d(剥,P)		○						
b(包,V)								
a(車,N)				○				○
f(る, P)			○					○
まで								
待							○	
つ								○
松								○

Summing up the algorithm, after morpheme dictionary consultation and the connection matrix, analyzing procedures are carried out as follows. As a notational convenience, words at column A, row B will be written in the abbreviation form Word(A,B).

- 1) Look up words in the dictionary and record the words to the CYK table. In the example shown in Table 1, words in the first column are recorded in the table.
- 2) Set variable L to 1.
- 3) Set variable L2 to 1.
- 4) Check whether there is a pair of words word₁(1,L) and word₂(1+L,L2) which can be concatenated under the constraints of the connection matrix. If there is a pair of words, record concatenated word to the well formed sub-string table as word_{new}(1, L+L2). If there is no pair of words, go to next.

- 5) Increment L2. If $1+L+L2$ is greater than the input length go to next, otherwise Go to 4).
- 6) Increment L. If $1+L$ is greater than the input length, go to next, otherwise go to 3).
- 7)end of the algorithm

After applying the algorithm above to the example, we will obtain the CYK table shown as follows. As a notation, (A;B) means A or B, and (A,B,C) means a sequence of words A,B,C.

Table 3 : An Example of CYK Table

6	w((v,r),Vs) x(((t;u),n),N)					
5	v(((t;u), o),Vs)					
4	t((s,i),P) u((a,l),P)					
3	a(車,N)					
2	b(包,V) s(((c;d),f),P)		i(まで,P)	k(デマ,N)	n(松,N)	
1	c(来,Vs) d(列,P)	e(流,Vs) f(る,P)	g(間,N) h(巻,Vs)	j(出,Vs) l(で,P)	m(間,N) o(待,Vs)	p(津,N) q(釣,Vs) r(つ,Vs)
	く 1	る 2	ま 3	で 4	ま 5	つ 6

1.3. Rules and Dictionary

In this section, all the algorithms are defined in a more formal approach[Sugimura 89a]. In the previous section, morphological analysis

was introduced by a declarative method in which a sequence of words which was given before the analysis is checked as to whether it is acceptable or not using rewriting rules. In this process, acceptance is validated with rewriting rules. Therefore, this declarative algorithm could have various kinds of implementation. In this section, the fundamental algorithm will be developed using "concatenation" and "Non Deterministic Finite State Automaton (NDFSA)". Although in the implementation, this algorithm is realized as a transducer [Nagao 83], this model enables us to catch the crucial point of this algorithm especially for implementation.

1.3.1. Rules for Concatenation

This section explains the rules for concatenation. In our approach, morphological analysis is performed by applying rules which restrict the proper concatenation of adjacent morphemes. Before going further, we will define the notation formally.

We define a Japanese string ($\text{string}_u:[n]$) as a finite sequence, $u:[n] \rightarrow A$, where n is a natural number denoting the length of u , and A is a set of Japanese characters. Thus, if $u:[n] \rightarrow A$ is a string and $n > 0$, for every $i \leq n$, $u(i)$ is some element of A . [Gallier]

A Japanese string as an entry of a lexical dictionary usually has lexical ambiguities; thus we define a function, 'get_entries', which obtains lexical entries corresponding to an input string. As shown in Formula 1.-1, each morpheme is a triple, $(\text{string}_u:[n], A, B)$, where A and B are the left- and right-hand features which restrict the words to be concatenated. We will also use S_{LF} to refer to the set of left-hand-side features of morphemes. The symbol $\leftarrow$ denotes the left-hand-side feature of morpheme u . Similarly, S_{RF} is the set of right-hand-side features of morphemes, and $\rightarrow$ denotes the right-hand-side feature of morpheme u .

Formula 1.-1

$$\begin{aligned} \underline{\text{get_entries}(\text{string}_u[n])} &\stackrel{\text{def}}{=} \text{string}_u[n] \rightarrow \{\text{morpheme}_{u1}[n] \dots \text{morpheme}_{uk}[n]\} \subset \text{Sentries} \\ \text{where } \text{Sentries} &\stackrel{\text{def}}{=} \{(string_u[n], A, B) \mid A \in S_{LF} \wedge B \in S_{RF}\} \end{aligned}$$

We will use expressions of the form $\text{morpheme}_{u:}[n]$ to denote elements of the set Sentries . It is sometimes convenient to refer to $\xleftarrow{u}$ and $\xrightarrow{u}$ by the functional expressions $\text{left}(\text{morpheme}_{u:}[n])$ and $\text{right}(\text{morpheme}_{u:}[n])$.

Restrictions on acceptable concatenations are expressed by the predicate C_{rule} . The definition is that two morphemes, $\text{morpheme}_{u:}[m]$ and $\text{morpheme}_{v:}[n]$ can be concatenated if the right context of $\text{morpheme}_{u:}[m]$ is related to the left context of $\text{morpheme}_{v:}[n]$ as shown in Figure 1.

$$\begin{aligned} &\left\{ \begin{array}{c} \overbrace{\left\langle \xleftarrow{u1} \text{morpheme}_{u1}[m] \xrightarrow{u1} \right\rangle}^{\text{string}_u[m]} \\ \vdots \\ \overbrace{\left\langle \xleftarrow{up} \text{morpheme}_{up}[m] \xrightarrow{uq} \right\rangle} \end{array} \right\} + \left\{ \begin{array}{c} \overbrace{\left\langle \xleftarrow{v1} \text{morpheme}_{v1}[n] \xrightarrow{v1} \right\rangle}^{\text{string}_v[n]} \\ \vdots \\ \overbrace{\left\langle \xleftarrow{vq} \text{morpheme}_{vq}[n] \xrightarrow{vj} \right\rangle} \end{array} \right\} \\ &\quad \Downarrow \\ &\quad \overbrace{\left\langle \xleftarrow{ui} \text{morpheme}_w[m+n] \xrightarrow{vj} \right\rangle}^{\text{string}_w[m+n]} \\ &\text{string}_w(i) = \begin{cases} \text{string}_u(i) & \text{if } 1 \leq i \leq m \\ \text{string}_v(i-m) & \text{if } m+1 \leq i \leq m+n \end{cases} \\ &\quad \text{where } C_{rule}(\xrightarrow{ui}, \xleftarrow{vj}) = T \end{aligned}$$

Figure 1: Concatenation of morphemes

In the traditional method, C_{rule} is defined as a proposition shown in Formula 1.-2.

The functions leftf and rightf are realized as functions shown in Formula 1.-2. Here, leftf and rightf will return a natural number. In [Yoshimura 83] the number is at most 400.

In this traditional method, the definition of $\text{Set}_{connect}$ is slightly hard for grammar writers to handle, since the same number occurring in sets S_{LF} ,

and S_{RF} will have different interpretations. An element in $Set_{connect}$ does not have anything that makes it easy to handle. However, in the LAX system, we define the functions $rightf$ and $leftf$, and predicate $Crule$ as follows.

Formula 1.-2

$$\begin{aligned} Crule(morpheme_u:[m], morpheme_v:[n]) &= T \\ \Leftrightarrow leftf(morpheme_u:[m]) \cap rightf(morpheme_v:[n]) &\neq \emptyset \\ rightf(morpheme_u:[m]) &\subset S_{feature} \\ leftf(morpheme_v:[n]) &\subset S_{feature} \end{aligned}$$

$S_{feature}$ in Formula 1.-2 is the set of morphological features which represents the relationships between morphemes. In other words, $Crule$ holds exactly when $rightf(morpheme_u:[n])$ and $leftf(morpheme_v:[n])$ have only one common element, as shown in Formula 1.-2.

In the case that the rules are taking the form of the above definition, the process of finding correct concatenation in LAX can be formalized as a non-deterministic finite state automaton as follows.

Formula 1.-3

$$M \stackrel{\text{def}}{=} (K, \Sigma, \delta, Q_0, F)$$

Formula 1.-4

$$\begin{aligned} K \stackrel{\text{def}}{=} \{X \mid X = (\{rightf\}, string_m:[n])\} \\ \text{where } \{rightf\} \subset S_{feature} \end{aligned}$$

Formula 1.-5

$$\Sigma \stackrel{\text{def}}{=} \{X \mid X = string_v:[n]\}$$

Formula 1.-6

$$\delta((q, \text{string}_a:[m]), \text{string}_x:[n]) \stackrel{\text{def}}{=} \bigcup_M (\text{rightf}(M), \text{string}_x:[M+N])$$

where $M \in \text{get_entries}(\text{string}_a:[n]) \wedge C_{\text{rule}}(q, \text{leftf}(M))$

Formula 1.-7

$$\delta(q, \epsilon) \stackrel{\text{def}}{=} (q, \text{string}_a:[0])$$

$$\delta(q, ax) \stackrel{\text{def}}{=} \bigcup_{p \in \delta(q, a)} \delta(p, x)$$

where $q \in K, a \in \sigma, x \in \sigma^*$

$$\delta(S, x) \stackrel{\text{def}}{=} \bigcup_{(q, \text{input_string}) \in S} \delta(q, x)$$

where $S \subset K, x \in \sigma^*, q \in F$

The NFSA " M " is defined in Formula 1.-3. K is the set of states of M , and σ is the set of alphabets for M . δ is the transition function. Q_0 is the initial state, and E is the final state.

K is the set of right restrictions as defined in Formula 1.-4.

σ is the set of Japanese strings that are in entries of the LAX lexical dictionary. The string look-up process in the lexical dictionary is realized as a TRIE structure, and takes a finite interval of less than 0.5 msec in the worst case. Therefore, we start our discussion assuming that Japanese strings in the dictionary have already been found.

The definition of the transition function, δ , is slightly complicated. If the Japanese string $\text{string}_a:[n]$ has more than one $\xleftarrow{a}$, then $\delta(q, a)$ is defined as shown in Formula 1.-6 and Formula 1.-7.

The definition of δ in Formula 1.-7 is the same as the usual definition for an NFSA.

In LAX, plural transition tables (TrTbl) can be used as shown in Figure 2. Therefore, the elements of S_{feature} are defined as follows.

$$S_{feature} \stackrel{def}{=} \{X | X = Name_of_TrTbl(State)\}$$

Where State is the name of a state in Trtbl

The multiple transition table is very convenient for representing rules of derivation and rules of inflection in a separate form. See [Morioka] and [Sano 88] for details.

Figure 2 shows the analysis of the Japanese word **bunsetsu** ``痛めるな" [ITAMERUNA] (do not hurt::痛める = to hurt, な = do not). In this figure, the input category  $I_{dm}$  of the word stem 痛 (word stem of the verb ``hurt") is accepted in the initial state S_{di} . Then, the next state is S_{di} or S_{dj} . The next input character is め. $leftf(め)$ is accepted in the state S_{di} and the next state becomes S_{dk} or F , that is the final state. The next character is accepted in the state S_{dk} and the next state becomes S_n or F . S_n is the state in the inflection Table. The last input "な" is accepted in the inflection table and the word ``痛めるな" is accepted.

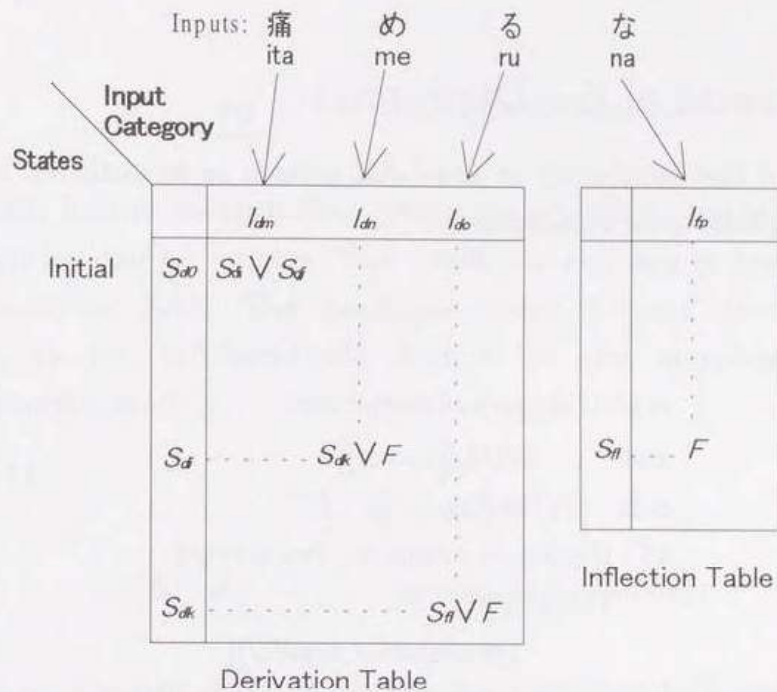


Figure 2 : An Analysis of a word by a multiple transition table

1.3.2. Expansion of NFSA

The LAX system also provides a "handle" which allows the lexicon-writer to build arbitrary data-structures in parallel with the analysis.

The arbitrary data structures can be attached by assigning the function shown in Formula 1.-8 to each element of $S_{feature}$. This function enables us to attach some data structures to $\leftarrow$ and $\rightarrow$, using them, we can write programs to construct morphological semantic information, which will be described in the following section.

Formula 1.-8

$$\text{info}(X) \rightarrow \{\text{some data structures}\}$$

where $X = \text{leftf}(\text{morpheme}_x[n]) \vee X = \text{rightf}(\text{morpheme}_x[n])$
 $\text{morpheme}_x[n] \in S_{entries}$

1.3.3. Format of the Dictionary

The syntax of the dictionary in the LAX system is as outlined below. Each entry has the following structures.

Formula 1.-9

```

begin(Category of morpheme)
surf  :: TrTbla([sta1,...])
&&   [TrTblb([Stb1,...]),...]
$$   [[in(Input), Common_Procedures],
      TrTblb(Stb1,
              [procedureb11, send(O)],
              [procedureb12, call_sax(O2 )])...
end(Category of morpheme)

```

Morphemes with the same category are defined in Formula 1.-9. The category name of morphemes is written in the first line and the last line,

In the second line, the surface character of a morpheme is written first. “::” is a delimiter. Then, *leftf(Morpheme)* is written. *leftf(Morpheme)* can be regarded as input to the NFSA. If the current state is *rightf(Morpheme_u)* = {動詞派生接辞め(word stem for verb “me”), ウ系強変化マ(word stem for verb “ma”) } and the input is *leftf(Morpheme_v)* = {動詞派生接辞め}, then string_v will be accepted according to the definition of Formula 1.-2.

In the third line, *rightf(Morpheme)* is defined.

From the fourth up to seventh lines, we can write programs which correspond to the next state transitions.

In the fourth line, common programs can also be written. Programs written in this part are activated when the morpheme is accepted. *in(I)* is a function to obtain information from the preceding morpheme. *In(I)* is the realization of the complex function *info(leftf(morpheme))* as shown in Formula 1.-8.

Formula 1.-10

$$\text{in}(I) \quad \underline{\text{def}} \quad I = \text{info}(\text{leftf}(\text{Morpheme}))$$

From the fifth line to seventh line, programs which correspond to the next transition states can be written. The predicate *call_sax* is used to call the syntactic analyser SAX. The predicate *send* is used to attach some information to the left-hand-side feature of the morpheme. *send* is *info(rightf(morpheme))*.

Formula 1.-11

$$\text{out}(O) \quad \underline{\text{def}} \quad O = \text{info}(\text{rightf}(\text{Morpheme}))$$

When $C_{rule}(\rightarrow, \leftarrow)$ is true, information on the right-hand-side and left-hand-side features is unified.

1.4. The Elementary Implementation

The elementary algorithm can be executed by multi-processes which are layered in the well formed sub-string table as shown in Figure 3.

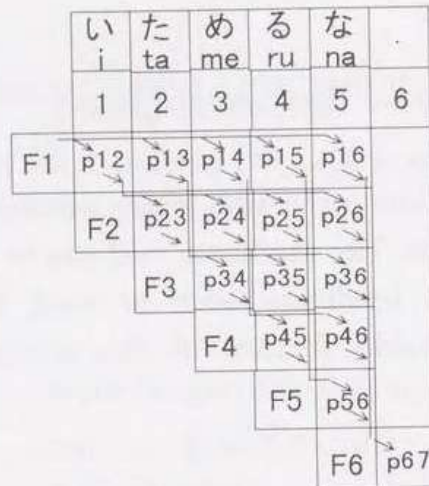


Figure 3 : Multi Processes in a well formed sub-string table

In Figure 3, the input string is 'いためるな'[i-ta-me-ru-na](do not fry). In this figure, a string of **Kana** is used as input to the LAX for expository reasons. The characters are numbered from 1(い) to 6(the punctuation mark '.').

p_{ij} represents a search process. F1 to F6 represent the row numbers of the table.

In the LAX dictionary, a process p_{RC} looks up morphemes corresponding to the string which starts from the character at R and ends at C-1. For example, process p_{23} looks up morphemes which have た as the surface. Search processes p_{12} to p_{67} can be executed simultaneously.

Each process is connected through streams (data paths) which the searched morphemes are passed through. Outputs from the processes at column C are merged to be the input to processes at row C. For example, outputs from p_{14} , p_{24} , and p_{34} are merged to be the inputs to p_{45} , and p_{46} ; p_{14} , p_{24} , and p_{34} look up morphemes corresponding to the string which ends at め, and p_{45} and p_{46} look up morphemes corresponding to the string which starts from る. Therefore, connecting these processes by a stream, we can concatenate morphemes.

An outline of the elementary algorithm is as follows.

1. Activation of processes

- (a) Activate $(N(N+1)/2)+1$ processes corresponding to an input string

which has N Japanese characters and a Japanese punctuation mark. $(N(N+1)/2)$ processes correspond to N Japanese characters. Activated processes are linked with streams which are indicated by arrows in figure Figure 3

(b) the initial state of the NFSA is put in the stream into the processes at row 1.

(c) Data from $p(N+1)(N+2)$ is the result of the analysis.

2. An activated process PNK then executes the following algorithm

(a) Wait for input Min.

(b) If the input data indicates the tail of the stream such that *nil*, output *nil* and stop. Otherwise, go to the next step.

(c) Look up morphemes Mrf which correspond to the string starting from N and ending at $K-1$. If Mrf exists, go to the next step. Otherwise stop.

(d) Call C_{rule} to check whether Min and Mrf can be concatenated or not.

If they can be concatenated, concatenate Mrf and Min, then output the concatenated morpheme. Go to 2(a).

This method makes it possible to perform parallel lexical analysis, although there are still some problems in practical use, as follows.

1. Although $((N+1) \times N/2) + 1$ processes will be activated in 1a, few of these processes will be able to look up morphemes Mrf at 2(c). Therefore, the efficiency of analysis will drop because of this useless activation.
2. The elementary algorithm cannot handle unknown lexical entries.
3. The elementary algorithm produces each result separately so that the worst complexity will be as follows.

Formula 1.-12

$$O = \sum_{i=0}^{n-1} a^{i+1} \times \frac{(n-1)!}{(n-1-i)!i!}$$

$$= a(1+a)^{n-1}$$

where a is the worst ambiguity of the morpheme and
n is the character length of the input sentence

Therefore there should be a packing algorithm. As described in section 5, this reduces the complexity to $O(n^3)$ as follows.

Formula 1.-13

$$O = \sum_{i=1}^{n-1} i \times (n-i) \times a^2$$

$$= \frac{a^2 \times (n^3 - n)}{6}$$

where a is the worst ambiguity of the morpheme and
n is the character length of the input sentence

$$\overbrace{\underbrace{C_1, C_2, \dots, C_i}_i, \underbrace{C_{i+1}, C_{i+2}, \dots, C_n}_{n-i}}^n$$

A practical algorithm which activates minimal processes, can handle the unknown lexical entries, and has complexity $O(n^3)$ will be described in the next section.

1.5. Effective Implementation

In this section, the practical algorithm will be represented. First efficient activation of the processes is represented. Second, the data flow with packing is described.

1.5.1. Making Processes Activation Demand-Driven

This subsection describes an efficient way to minimize avoidable activation of processes. At the beginning of the analysis, a “dict” process is activated for each input character as shown in Figure 4.

	い	た	め	る	な	.
	i	ta	me	ru	na	.
	1	2	3	4	5	6
F1	dict	p13	p14			
F2		dict				
F3			dict	p35		
F4				dict		
F5					dict	
F6						dict

Figure 4 : An Example of a “dict” processes corresponding to input characters

$N+1$ “dict” will be activated when an input string has N characters and a punctuation mark. (In Figure 4, six “dict” processes were activated.) “dict” looks up morphemes which correspond to the string from the point at which “dict” was located. (For notation, we use function $\text{positp}(\text{dict})$ to indicate this point.) For example, the “dict” process at F1 in Figure 4 looks up morphemes corresponding to strings which start from い[i] in the first column. If there are morphemes corresponding to いた[i-ta] in the dictionary, the “dict” at F1 will activate sub-process P13. Similarly, if いため[i-ta-me] is in the dictionary, the “dict” at F1 will activate sub-process P14. If there is no

morpheme corresponding to いためる [i-ta-me-ru], no process is activated. For convenience, the notation *refmorph(dict)* or *refmorph(PXY)* will be used in this paper to represent a set of morphemes corresponding to the sub-string to which “dict” or PXY are attached. A “dict” process and its sub-process can be obtained as programs by compiling the lexical dictionary of the LAX. For example, let us consider the tiny lexical dictionary shown in Table 4. This can be represented as a TRIE structure as shown in Figure 4, and then compiled into a “dict” program, as shown in Figure 6. To simplify the representation, Figure 6 shows only the look-up processes.

Table 4 : An Example of a Lexical Dictioanry

Surface	Category
い[i]	Word stem of the noun 胃 (stomach)
いた[ita]	Word stem of the adjective 痛 (sore)
いた[ita]	Word stem of the verb 傷 (to hurt)
いため[itame]	Word stem of the verb 炒 (to fry)
いため[itame]	Word stem of the noun 板目 (wood grain)
いたく[itaku]	Word stem of the nominal verb 委託 (trust, to entrust)

```

[い, word-stem-of-noun(胃),
  [た, word-stem-of-adjective(痛),
    word-stem-of-verb(傷),
    [め, word-stem-of-verb(炒め),
      word-stem-of-noun(板目)],
    [く, word-stem-of-noun-verb(委託)]
  ]
]

```

Figure 5 : TRIE Structure for a Lexical Dictionary

```

dict(い,R,I) :- R = [H|RR],
  I = [[word-stem-of-noun(胃)]|Ir],
  い(H,RR,Ir).
いた(た,R,I) :- R = [H|RR],
  I = [[word-stem-of-adjective(痛)],
    [word-stem-of-verb(傷)]|Ir],
  いた(H,R,Ir).
い(_,_,[]).
いた(め,R,I) :- I = [[word-stem-of-verb(炒め)],
  [word-stem-of-a-noun(板目)]].
いた(く,R,I) :- I = [[word-stem-of-noun-verb(委託)]].
  いた(_,_,[]).

```

Figure 6 : An Implementation of a Dictionary on Prolog

In Figure 6, the first argument of “dict” is the character to which “dict” corresponds. The second argument is the next sequence of input characters. The last argument returns the result of the search.

Let us see what happens when we enter the sequence [い,ほ,め,る].

The “dict” corresponding to the first character “い” will accept this sequence. Therefore, “dict” will obtain the word-stem-of-noun(胃). Next, “dict” calls い([ほ,め,る],RR,Ir), and this call does not bring any more results, so that no redundant processes which search sub-strings such as [い,ほ] or [い,ほ,め] or [い,ほ,め,る] will be activated.

With regard to the efficiency of the search, the program in Figure 6 is fast because logic programming languages like Prolog search for clauses using first argument hashing. The TRIE structure enables us to utilize this hashing mechanism easily, making the "dict" process perform its search rapidly.

As stated above, compilation of the lexical dictionary enables us to minimize the number of activated processes, and to look up morphemes quickly.

1.5.2. Data and Its Flow in the Practical Algorithm

This section describes the data flow between "dict" and its sub-processes, thereupon presents a data packing mechanism which makes the complexity of the practical algorithm $O(n^3)$.

Data flow

The "dict" processes described in section 1.5.2. are connected with each other through data streams as shown in Figure 7.

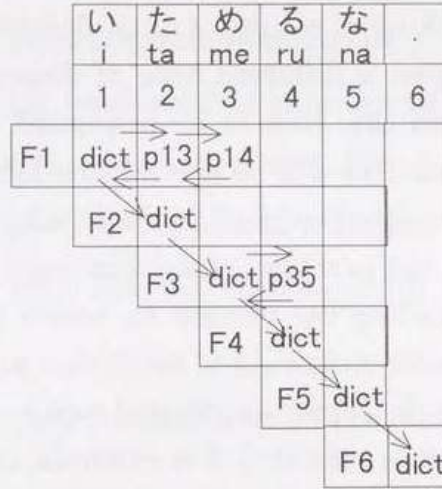


Figure 7 : "Dict" Processes connected by Data Streams

The data stream structure is displayed in Figure 8. This structure illustrates the trace of morphological analysis, and can be regarded as an application of the layered stream technique.

Unlike SAX, which analyses inputs according to CFG with the layered stream technique, LAX does not place any identifier between the rules.

<data stream>	:=	[<array of data>]
<array of data>	:=	<data>,<array of data>
<array of data>	:=	<data>,
<data>	:=	[<position>, <r-features>, <results>]
<position>	:=	$N(1 \leq N \leq \text{length of input})$
<r-features>	:=	[<re>,...,<re>]
<re>	$\in$	S_{feature}
<results>	:=	[<surface of a morpheme>, <morphological category>, <array of results>]
<array of results>	:=	[]
<array of results>	:=	<results>, <array of results>

Figure 8 : Definition of The Data Stream Structure

In Figure 8, $\langle r\text{-features} \rangle$ corresponds to $rightf(\text{morpheme})$. $\langle \text{array of results} \rangle$ enables the packing of resultant data, described in the next section. In the practical algorithm, a different type of data will flow into a “dict”, which should distinguish the data to be processed in it. For example, in Figure 7, the results from P14 will be passed through the “dict” in lines F1, F2, and F3 before flowing into the “dict” in F4, in which the results should be processed. Therefore, in the practical algorithm, each item of data has a tail position number representing the process by which it should be processed. (We use the function $position(data_{data})$ to mark this position.) Corresponding to $position(data_{data})$, each “dict” is tagged with the position number $position(dict_{process})$ when it is activated. For example, in Figure 7, the “dict” in line F1 is tagged with 1 and the “dict” at line F3 is tagged with 3 at the beginning of the analysis.

Let us trace through the example computation in detail. First, the token $[1, [end(top)], []]$ is given to the “dict” in line F1. This can be regarded as initializing the NFSA.

The “dict” at F1 will output the token $[2, rightf(\text{胃}), [\text{胃}, \text{word-stem-of-noun}, []]]$ after calling C_{rule} to check whether top and the word-stem-of-noun(胃) can be concatenated.

The “dict” in F1 activates P13, and P13 activates P14 in succession. Then, the data stream coming out from the “dict” in F1 into the “dict” in F2 is now as shown in Figure 9.

```
[
  [2,[right{胃}],      [胃,word-stem-of-noun,[]]],
  [3,[right{痛}],      [痛,word-stem-of-adjective,[]]],
  [3,[right{傷}],      [傷,word-stem-of-verb,[]]],
  [4,[right{炒め}],    [炒め,word-stem-of-verb,[]]],
  [4,[right{板目}],    [板目,word-stem-of-noun,[]]]
]
```

Figure 9 : An Example Data Stream

Concatenation with Data Packing

On receiving the inputs, as shown in Figure 9, the “dict” in F2 picks up data of whose *position(data_{data})* is 2, and outputs data whose *position(data_{data})* is greater than 2 to the “dict” in line F3. The “dict” in F2 will check whether the data picked up can be concatenated with a morpheme corresponding to character strings starting from point 2. If there is no such morpheme, the “dict” in F2 will output [] and stop. Otherwise, the “dict” in F2 will work like the “dict” in F3.

The “dict” in F3 inputs data as follows.

```
[
  [3,[right{痛}],      [痛,word-stem-of-adjective,[]]],
  [3,[right{傷}],      [傷,word-stem-of-verb,[]]],
  [4,[right{炒め}],    [炒め,word-stem-of-verb,[]]],
  [4,[right{板目}],    [板目,word-stem-of-noun,[]]]
]
```

Let us suppose that, using the “dict” in F3, a morpheme め is found as follows.

Formula 1.-14

$$\begin{aligned} \text{leftf}(\text{inflect}(\text{め})) \cap \text{rightf}(\text{word_stem_of_adjective}(\text{痛})) &\neq \phi \\ \text{leftf}(\text{inflect}(\text{め})) \cap \text{rightf}(\text{word_stem_of_verb}(\text{傷})) &\neq \phi \end{aligned}$$

Then, the output from the “dict” in F3 will be as follows.

```
[
  [4,[rightf{め}],[め,inflect,
    [  [痛,word-stem-of-adjective,[]],
      [傷,word-stem-of-verb,[]]]],
  [4,rightf{炒め},[炒め,word-stem-of-verb,[]],
  [4,¥rightf{板目},[板目,word-stem-of-noun,[]]]
]
```

Figure 10 : Packed Data

The result shown in Figure 10 is very important. In LAX, combinatorial ambiguities which occur in concatenation are not propagated to any other concatenation. If there are ambiguities, they are packed as shown in Figure 10. Therefore the complexity becomes $O(n^3)$. Details of complexity will be discussed in section 1.5.3..

Summary of the Effective Algorithm

This section summarizes the practical algorithm.

1. Activation of “dict” processes

- (a) Activate $N+1$ “dict” processes corresponding to each input strings which has N Japanese characters and a Japanese punctuation mark. Each “dict” should have a *position(dict_{process})*. N “dict” processes correspond to N Japanese characters. Activated processes are linked with streams in the way indicated by the arrows in Figure 7.

- (b) *rightf(top)*, indicating the initial state of the NFSA, is put in the stream to the processes at row 1.
- (c) Data from the “dict” at $N+1$ is the result of analysis.

2. Algorithm for the “dict” processes

The “dict” processes activated in lines 1 continue as follows. A “dict” has many sub-processes, which are connected with each other through communication streams, as shown in figure Figure 11. Each sub-process can be characterized by its *input* and process and its *output* to its connected processes. From this line, the function *input(Data)* represents the data input of each process, and the function *output(Data, Process_N)* represents the data output to the *Process* whose number is N . Numbers in Figure 11 correspond to the following item numbers.

(a) Do input(Data)

If *Data* is the *tail* of a data stream, do *output(tail, Process_{2b})* and stop. Otherwise, do *output(Data, Process_{2b})*. Go to 2a.

(b) The *filstr* process.

Do *input(Data)*.

If *Data* is the *tail* of a data stream, do *output(tail, Process_{2c})*, *output(tail, Process_{2d})*, *output(tail, Process_{2e})*, and stop.

If *position(Data_{data})* is equal to *position(Dict_{data})* do *output(Data, Process_{2c})*, and *output(Data, Process_{2d})*,
If *position(Data_{data})* is greater than *position(Dict_{data})*, do *output(Data, Process_{2e})*, Go to 2b.

(c) The *filcon* process.

Input(Data).

Check whether *C_{rule}(Data, refmorph(dict))* is true or not.

If it is true, concatenate *Data* and *refmorph(dict)* to make new morphemes and *output(new morphemes, Process_{dict:pro:5})*. Go to 2c.

(d) The PXY sub-process

i. Do *input(Data)*.

If *Data* is the *tail* of a data stream, *output(tail, Process_{2(d)ii})* and stop.

Otherwise, do *output(Data, Process_{2(d)ii})*, recursively call PXY and do *output(tail, Process_{dict:pro:4})*. Go to 2(d)i.

ii. Do *input(Data)*.

If *Data* is the *tail* of a data stream, do *output(tail, Process_{2(d)ii})* and stop.

Check whether $C_{rule}(Data, refmorph(sub_pro))$ is true or not.

If it is true, concatenate *Data* and *refmorph(sub_pro)* to make new morphemes and do *output(new morphemes, Process_{dict.pro:5})*

Go to 2(d)ii.

(e) Do *input(Data)*.

If *Data* is the *tail* of a data stream, and it is the first input data, process the unknown word, else do *output(Data, Process_{thenextdict})* and stop.

Otherwise, do *output(Data, Process_{thenextdict})*.

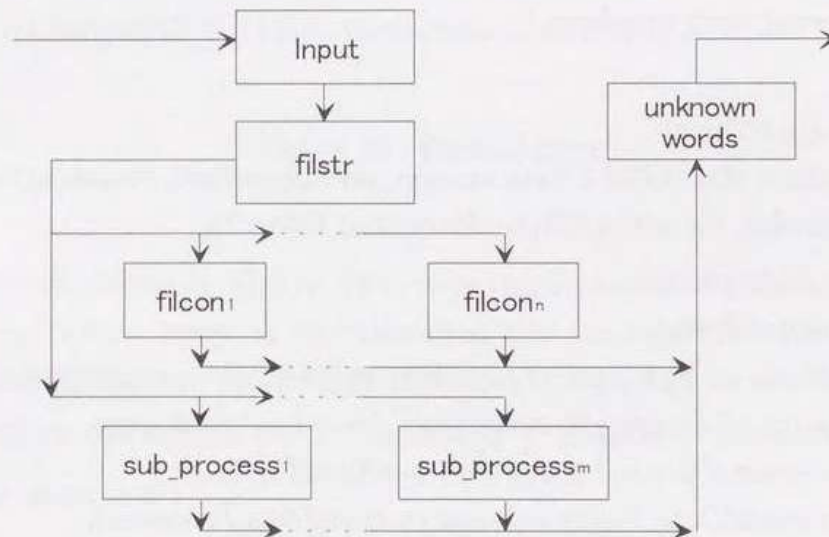


Figure 11 : Process Layout for Lexical Analysis

1.5.3. Complexity

This section discusses the complexity of the elementary and practical algorithms, in sequential and parallel execution.

Complexity in Sequential Execution

First, the elementary algorithm has the following complexity. In the following expressions n is the number of input characters, and T is the total amount of time required to obtain all solutions, so that complexity is that of T related to n . We assume that any input sub-string corresponds to a morphemes. All adjacent morphemes can be concatenated. d is the time required for dictionary search. c is the time required to check and concatenate a pair of morphemes. I is the number of delimiting positions as shown in the chart under the expressions. $a^{i+1} \times_{n-1} C_i$ is the time required to check and concatenate all the pairs of morphemes when there are i delimiters in character strings.

Morpheme look-up is realized with the TRIE structure, and the search time for a morpheme of m characters takes $d \times m$ where d is the time required to trace the TRIE structure. Thus, the total look-up time for sub-string contains j input characters, $d \sum_{i=1}^n i(n-i+1)$. Thus we have

Formula 1.-15

$$T = c \sum_{i=0}^{n-1} (a^{i+1} \times_{n-1} C_i) + \sum_{j=1}^n dj(n-j+1)$$

$$= c \times a(1+a)^{n-1} + d \frac{n(n+1)(n+2)}{6}$$

where $a, n \geq 1, c, d > 0$

$$O(T) = O(a^n)$$

and thus,

$$\overbrace{C_1, C_2, \dots, C_a}^{\text{morpheme}_1}, \overbrace{C_{a+1}, \dots, C_b}^{\text{morpheme}_2}, \dots, \overbrace{C_{m+1}, \dots, C_n}^{\text{morpheme}_{i+1}}$$

There should be a limitation on the morpheme length so that the maximum character length of morpheme can be set to l . If l is greater than n , the complexity becomes $O(a^n)$. Otherwise, it becomes as follows. It is impossible to get one expression for Formula 1.-16 because it is impossible to obtain any expressions for polynomials whose maximum degree is " n ".

Formula 1.-16

$$T = cA_n$$

where

$$A_k = \sum_{i=1}^l A_{k-i} (k > l)$$

$$A_k = ca^k (k \leq l)$$

where $c > 0, k > l$

Then the complexity of the practical algorithm is as follows:

Formula 1.-17

$$T = c \sum_{i=1}^{n-1} (i \times (n-1) \times a^2) + \sum_{j=1}^n dj(n-j+1)$$

$$= \frac{a^2 \times (n^3 - n)}{6} + d \frac{n(n+1)(n+2)}{6}$$

$$O = O(n^3)$$

where a is the worst ambiguity of the morpheme and
 n is the character length of the input sentence

$$\overbrace{C_1, C_2, \dots, C_i, C_{i+1}, \dots, C_n}^n$$

$\underbrace{\hspace{10em}}_i$
 $\underbrace{\hspace{10em}}_{n-i}$

If the maximum length of the morpheme is i , the complexity is given by:

Formula 1.-18

$$\begin{aligned}
T &= c \sum_{i=1}^n a^2 + d \sum_{j=1}^l j(n-j+1) \\
&= c \times a^2 \times n + d(1+l) \times l \times \frac{1}{2} \times n + K \\
\text{where } K &= -d \sum_{j=1}^l j^2 + \sum_{i=1}^l \\
O &= O(n)
\end{aligned}$$

The conclusion to be drawn is clearly that Packing is critical to the algorithm.

Complexity in Parallel Execution

We define the time complexity of a parallel execution in terms of the time interval from the beginning of the parallel execution until its end. In parallel execution, dictionary search processes can run simultaneously, so that search time is proportional to the maximum length of morphemes. However, the concatenation processes cannot run simultaneously, since a process suspends itself until its preceding process outputs data to it. Therefore, in parallel execution, time complexity will be proportional to input length.

1.6. Heuristics for Disambiguation

In order to implement the algorithm for machine translation systems, heuristics to select the most appropriate result were implemented [Nakajima]. This algorithm enables us to select the best path in sequential time. In this section, the effective heuristics will mainly be presented.

Lets consider the following example.

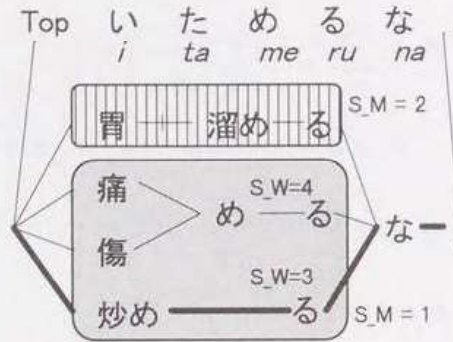


Figure 12 : An Example of Results of Lexical Analysis

In Figure 12, there are four outputs corresponding to the input “いためるな (*i ta me ru na*)”. In such a case, two principles for prioritization can be applied as follows. As a notational convention, $S_M(A)$ means an output which has a number of “A” morphemes in which word stems, and their derivational words are included, $S_W(B)$ means an output which has a number of “B” words in it, and $P(S)$ is a function to get a priority number for the output S .

$$(H1) \quad P(S_W(A)) > P(S_W(B)) \Leftrightarrow A < B.$$

$$(H2) \quad P(S_M(A)) > P(S_M(B)) \Leftrightarrow A < B.$$

At first, the heuristic H1 is applied to prioritize the outputs. Then, heuristic H2 will be applied to them.

In order to apply these two heuristics to the solutions, the number of words needed to make the solution is calculated during the analysis process as shown in the following example. In the example, we utilize a special notation to note the tentative result of analysis as follows.

$$\text{Entry} = N[C, L, S_W, S_M]$$

N : Identifier of each entry of analysis.

C : Morpheme

L : a list of morphemes which are connected to a morpheme C on the left hand side.

S_W : Number of words which appear on left hand side of the morpheme C .

S_M : Number of morphemes which appear on the left hand side of the morpheme C .

The Table 5 represents an analysis of the example shown in Figure 12.

Table 5 : An Example of Resultant Data Structure of Lexical Analysis

position	Char	
0		0[top]
1	い	1[胃, [0], 1, 1], 2[痛, [0], 1, 1], 3[傷, [0], 1, 1], 4[板目, [0], 1, 1], 5[炒め, [0], 1, 1]
2	た	6[溜め, [1], 2, 2],
3	め	7[め, [2, 3], 1, 2],
4	る	8[る, [7], 1, 3], 9[る, [5], 1, 2], 10[る, [6], 2, 3]
5	な	11[な, [9], 1, 3]
6	.	12[. , [11], 2, 4]

In the analysis, morphemes which start from position 1 are registered to the table. They are the morphemes 1, 2, 3, 4, 5.

Then, morphemes which begin from position 2 are looked up and checked for whether they can be concatenated with morphemes which end at position 1. Fortunately there is one morpheme “1[胃, [0], 1, 1]” which can be concatenated with a morpheme 6[溜め, [1], 2, 2], therefore the identifier of “1[胃, [0], 1, 1]”, 1, is registered to the morpheme “6” as the left hand side morpheme which can be concatenated to morpheme “6”. In this case, the number of words is incremented from 1 to 2, and also the number of morphemes is also incremented.

Next, morphemes beginning from position 3 are looked up and checked for whether there are some morphemes to be concatenated. In this case, there are two morphemes “2” and “3” which can be concatenated to the morpheme “7” which begins from position 3. Therefore, a new entry 7[め, [2, 3], 1, 2] is registered. Also the number of morphemes is incremented from 1 to 2.

Afterwards, morphemes from position 4 are looked up from the dictionary and checked for whether they can be concatenated with morphemes which are already registered to the well-formed sub-string table. There are three morphemes 8, 9, and 10 which can be concatenated with the morphemes registered. The number of morphemes and words is calculated.

Then, morpheme 11, which begins at position 5 is looked up. In this case, there are three morphemes which can be concatenated with the morpheme 11, but morpheme 9 has the least number of constituent words, therefore

morpheme 11 is concatenated only with morpheme 11. Then we can get a sequence of morphemes which follows the heuristics H1 and H2.

It is important to note that this process can be carried out in a fixed amount of time in each position. Therefore, time complexity is proportional to the length of the input.

1.7. Implementation on Logic Programming Language

As to the implementation on sequential and parallel languages, crucial issues are the realization of basic algorithms using characteristics of both languages.

1.7.1. Implementation on Prolog

Prolog facilitates a hash indexing mechanism which makes a hash table from each predicate name and the first argument name of the predicate. Therefore, I concluded that an effective implementation of the dictionary in Prolog to be as shown in Figure 6. An example of the implementation will be shown in appendix A. In the example, the process layout shown in Figure 11 is effectively used to realize the algorithm. This implementation could be used also in a parallel logic programming language. In a parallel logic programming language like KL1, the cut symbol becomes a committing operator.

One thing to add to the implementation issues is the connection checking which is realized by the Prolog predicate "check" in appendix A. This implementation aims at being easily understandable therefore I utilized an easy name "check". However, in the actual implementation of the connection matrix, we change the definition of the predicate check as follows.

```
:- check(A,B)
check(morph_name1, X) :- !, morph_name1(X).
check(morph_name2, X) :- !, morph_name2(X).
```

```

-----
check(morph_nameN,X) :- !, morph_nameN(X).
morph_name1(morph_a).
-----
morph_name1(morph_z).
morph_name2(morph_k).

```

As I explained before, a predicate and its first argument are utilized to make a hash table to retrieve the predicate quickly. Therefore, this mechanism is used to retrieve the connective relation as shown in the example above. In the example, when the predicate "check" was called, a first argument name is used to consult the connection relation table which has the first argument name as a left-hand side morpheme. Then, next to the direct call of the predicate `check(left, right)`, connectivity is checked with the predicate whose name is the first argument of the predicate "check" and whose argument is the second argument of the predicate "check" like `morph_left(morph_right)`. Such kind of multiple step retrieval for a matrix contents is a special device in logic programming language.

1.7.2. Implementation on PSI

Experimentally, we made a LAX system on PSI machine.

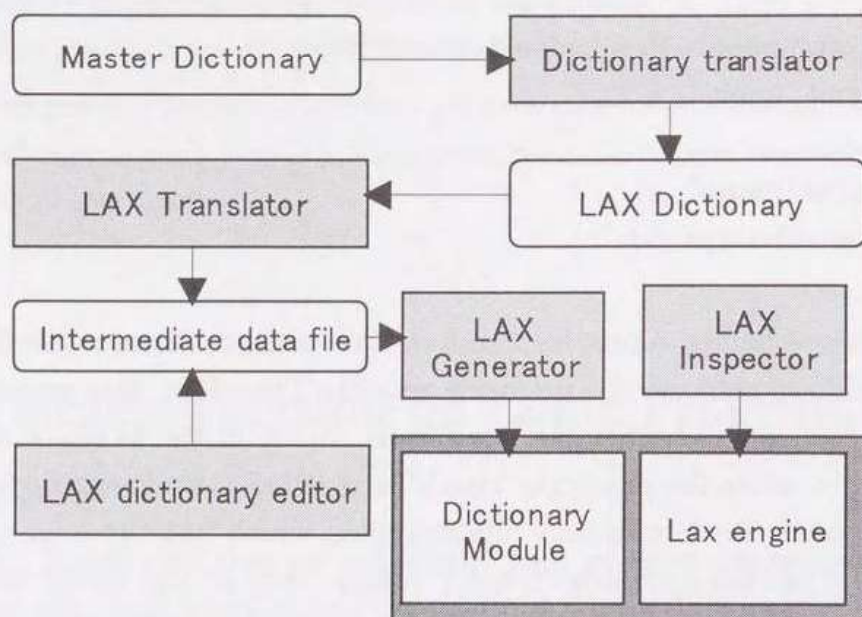


Figure 13 : LAX System on PSI Machine

Figure 13 shows the configuration of the LAX system[Kubo 88]. The dictionary translator transforms the master dictionary into the LAX dictionary. Using the LAX translator, the LAX dictionary is transformed into an intermediate data file. The analysis program is generated by the LAX generator from this intermediate data file. From the user's point of view, this intermediate data file is identical to the LAX dictionary.

To maintain the LAX dictionary, the LAX inspector provides many pieces of useful information. It has two modes: the analysis mode and the inspection mode. In the analysis mode, you can check the results of analysis, looking at the time and the number of ambiguities for each input sentence. If the analysis fails, you can also see which morpheme cannot be connected with its adjoining morpheme. In the inspection mode, you can also see which morpheme cannot be connected with its adjoining morpheme. In the inspection mode, you can examine the contents of the dictionary of the morphemes that belong to each adjoining point of input sentences.

We can easily modify the definitions of morphemes and add new entries by using the LAX dictionary editor. To modify a definition of a morpheme, we only have to rewrite the left-hand or right-hand features or semantic rule in

the edit window after locating the definition. To add new definitions, we may pick up another definition, copy it, and rewrite it.

In the case of the implementation on the PSI machine, we have 2000 morphemes for the purpose of the development of the discourse understanding system which will be introduced later in this thesis.

The LAX system has strong relation to the implementation of the system onto the workstation or personal computers. We pay meticulous attention to distinguish the description of the knowledge about morphological analysis from the implementation issues, and this attitude is important and effective when we try to implement the algorithm on the actual system.

1.8. Implementation on C

The algorithm presented in the previous sections have been implemented on workstations and personal computers. The implementation language is C. Particular issues in the implementation will be presented in the following section.

1.8.1. Implementation on a Workstation

In the implementation of the algorithm on the workstation, one of the most substantial issues is the implementation of the dictionary because there is no automatic hash index mechanism which is provided by Prolog or ESP which uses the predicate name and its first argument for the retrieval of predicates. The requirement to the dictionary retrieval is to look up a set of words which can match given sub-strings as shown in the example as follows. In the example, the given string is “つうかんこう [tsu-u-ka-n-ko-u]”, and the retrieval process of the dictionary should find a set of words which can match sub-strings which starts from a head of the given string such as “つ [tsu]”, “つう [tsu-u]”, “つうか [tsu-u-ka]”, “つうかん [tsu-u-ka-n]”, “つうかんこ [tsu-u-ka-n-ko]”, and “つうかんこう [tsu-u-ka-n-ko-u]”. In the case of the example, five words are found.



Figure 14 : An Example of Retrieval of sub-strings from a Dictionary

In order to realize the requirement, we designed a dictionary look-up procedure which utilizes “first key character index” and TRIE table on the main memory. This mechanism is shown in Figure 15.

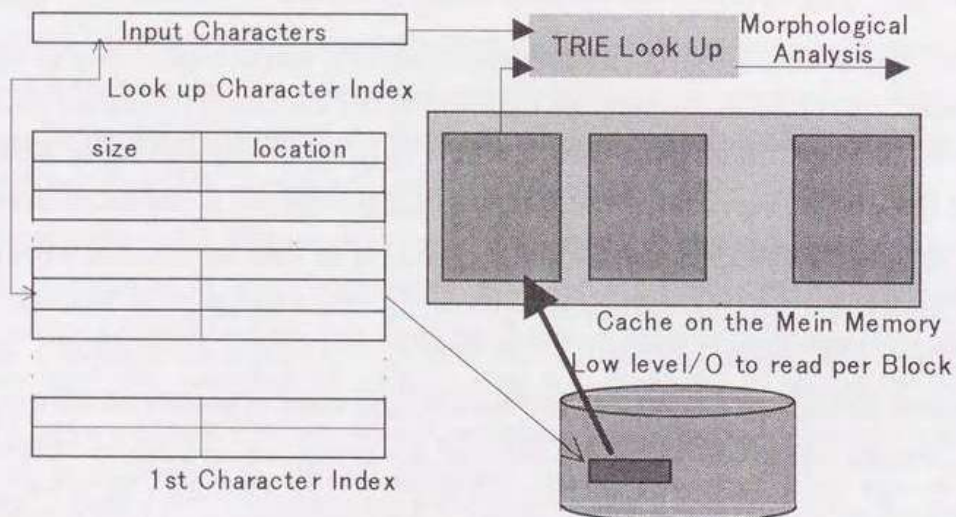


Figure 15 : The Retrieval System using Cache for TRIE Structures

This retrieval mechanism is carried out as follows.

- 1) Get a starting point in the input characters to retrieve the dictionary.

- 2) Fetch a character code which is at the starting point in the input characters.
- 3) Check whether a TRIE table which has the character of the starting point is at a cache memory. If there is already the TRIE table on the cache memory, go to 7), else go to next.
- 4) Calculate an address in the first character index from the character code which is at the starting point in the input characters.
- 5) Fetch the size of TRIE table and a location from the first character index.
- 6) Load the TRIE table from the hard disk into the main memory.
- 7) Carry out TRIE table look up and store the result to the CYK table.

Without this implementation of the dictionary retrieval, more than half the time of the translation process was occupied by the morphological analysis, mainly by the dictionary retrieval.

Although the caching mechanism is very simple, it dramatically improve the retrieval speed. In our experience, only 10 pages for caching memory reduces the morphological analysis speed by 20 percent on average.

Another matter concerning the implementation on the workstation is retrieval of the connection matrix. In the implementation of the algorithm in Prolog, such kind of connection matrix can be implemented easily because the ordered pairs of column and row numbers can be written in the predicate and its argument style as follows.

`Lefthand_side_morpheme(Righthand_side_morpheme).`

Though we can utilize such kind of powerful function in Prolog, in a procedural language like C, there should be a compiler which makes a connection matrix to realize fast retrieval.

1.8.2. Implementation on a PC

In order to implement the algorithm on personal computers, no special attention needs to be paid concerning development because the latest personal computers are equipped with many effective development tools. Almost all the code written in C can be compiled on the PC.

Therefore, main issues in the implementation on a PC were low level issues such as block size to transfer the TRIE page into the main memory. Before we could use an effective operating system which carries out complete resource management on memory, we did not try to implement the algorithm on the PC, because block size was limited to 64Kbytes, and this size is too small for the TRIE page.

After getting an effective operating system, we can easily implement the algorithm on WS to the PC. Basically there is no difference between the source code for WS and PC. Only performance in analysis speed differs.

1.9. Evaluation

1.9.1. Quantitative and Qualitative Evaluation

Although we need consistent evaluating criteria for the analysis speed between the implementation on Prolog/ESP, and C on a workstation and C on a personal computer, because there is a time difference among the developments, the evaluation methods were not consistent and differed, but gradually improved. Therefore, there could be some defects in the quantitative and qualitative evaluation because we could not utilize the same test set. However I believe these measurements will have some sense for the evaluation of the system performance. The test sentences shown in Appendix B were used to evaluate the first version of the LAX system on PSI machine. It is impossible to list up a whole test set of algorithms on a workstation and a personal computer. The test set on a workstation mainly utilized a set of sentences made in-house. The test set on a personal computer mainly utilized sentences withdrawn from home pages of the Internet.

Together with this collection of test sentences, we made an evaluation tool. The evaluation tool was made to evaluate the result and feed back the result to the researcher in charge of quality of the system.

For the purpose of evaluating precision and recall, we made a set of target results. These results were read by the evaluation tool, and compared with the result of analysis. This evaluating facility is designed as follows.

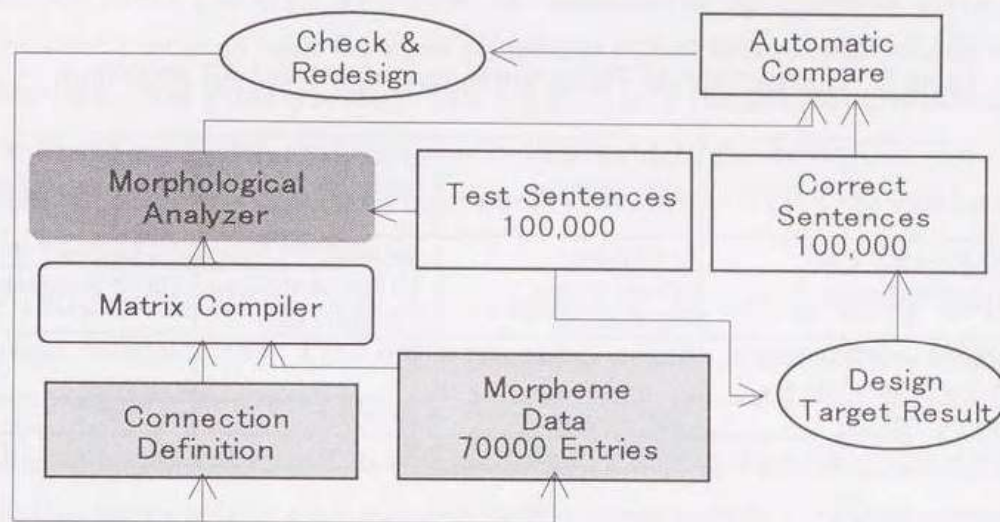


Figure 16 : The Evaluation System of the Morphological Analyzer

In Figure 16, test sentences of about 100,000 entries were collected from Internet home pages. This sentence resource is roughly analyzed by the morphological analyzer and checked, to make a target result, by the designer of the morphological analyzer. The result of the morphological analyzer is compared automatically with the target result and incorrect results are reported to the designer.

One of the most important points of this system is the Connection Definition part drawn at the bottom left of the figure. In this definition, the user inputs the ordered pair to display a connection relation between morphemes. There is no need to produce and maintain a connection matrix. The matrix compiler reads a sequence of the ordered pairs, and compiles them into a connection matrix. Using this evaluation tool, we could improve the quality of our morphological analyzer to achieve high precision as shown in Table 6 .

Table 6 : Comparison of Three implementations of LAX Algorithm

	ESP on PSI MM:320MB	C on WS(Sun-3) MM: 32MB	C on PC-AT Pentium 120Mz MM: 80MB
Analysis time per char	0.73msec	10 msec	12 msec
Evaluation Sentences	30sentences. 1370 chars	10,000 sentences 322,000 chars	100,000 sentences
Derivational and Inflectional morphemes, case markers	300 morphemes	600	1300
Word stems, nouns	500 entries	40,000	70,000
Precision	90	95	99.84

The main reason why the algorithm on PSI is faster than the algorithm on the workstation is the fact that on PSI, every dictionary is located in the main memory, but implementation on the workstation and the PC locates their dictionary on the hard disk.

As to qualitative evaluation, I will list up a set of example words which the system can not analyze properly in the following table.

No.	Patterns	Examples
1	Same surface and reading but with different POS	ない[nai] (Adjective or post positional marker) で[de] (Verb stem or post positional marker) もっ (Verb stem of 持つ or Verb stem of 盛る) とき (Verb “解き” as solve or noun “時” as time)
2	Same surface and different reading with different POS	勝った (Verb stem to read “Katta”) or 勝った (Verb stem to read “Masatta”), 降り (Verb stem to read “Ori”) or 降り (Verb stem to read “Furi”)
3	noun + noun or noun	時+雨 (Time / Rain) or 時雨 (shower)
4	noun + noun with different segmentation	一晩中(allnight)+火(fire)+を or 一晩(a night)+中火(middle fire)+を
5	Mixed pattern	むき出し(naked)+ <u>になった</u> (carry) or むき出し(naked) + <u>に</u> (post positional marker) + <u>なった</u> (become)

There are many patterns which can be classified to the patterns listed above in the first row. It is easy for us to eliminate one of morphemes which causes ambiguities, but this treatment will not give us a fundamental solution.

As to the second pattern listed above, it is hard to disambiguate the results without information about the dependency structure. This type will be solved using case agreement relations.

The third pattern has very few examples. I do not currently have any solution toward this kind of ambiguity. Some information related to discourse may give a solution.

With the fourth pattern, it is also very hard to solve the ambiguities.

As to the fifth pattern as a mixed pattern, some simple heuristics could give us a solution I think.

In any case, the word patterns listed above should be studied in future research on incorporating the technology which binds the morphological analyzer with the sentence analyzer.

1.9.2. Discussion

Fortunately, my work on LAX achieved good feasibility. The grammar framework was enriched and published from ICOT as freeware.

As for future research, I think the morphological analyzer should be merged with a parsing system which decides the relation between words. From this point of view, LAX has the promising function of being able to be connected easily to some parser. A promising work is the SAX parsing system which can input the packed data structure, which is a form of output of LAX, naturally. In fact, in DUALS, the LAX system was directly connected to SAX and played an important role in sentence analysis.

One of the hardest remaining problems is how to get information which is usually treated in sentence analysis into the morphological analysis. This kind of problem typically appears in honorific sentences as follows [Sugimura 86a] [Mizutani 1983].

Let us consider the sentence "Taro sama ga Hanako sama ni o megumi ni natta (Taro met Hanako)". In this example, there are interpretations as shown in the formula as follows. In the formula, " $A > B$ " means A is socially or mentally higher than B.

Taro <u>sama</u>	⇒	speaker < Taro
Hanako <u>sama</u>	⇒	speaker < Hanako
<u>megumi</u>	⇒	Taro > Hanako
natta	⇒	speaker > hearer

In such a case, usage of post positional markers displays the discourse situation such as social relations among speaker, hearer, and agents who appears in the sentence. These features tend to be treated as trivial matters in traditional natural language systems, but if we seriously take these features into account, we should make a morphological analyzer in which we can easily write not only the morphological rules, but also syntactic and semantic rules, and discourse rules as shown in the example above [Sugimura 86b]. These issues are partially studied after Fifth Generation Computer Symposium 88 (FGCS 88 for short) by developing a morphological analyzer which treats syntactic and semantic information together [Kubo 89]. As a direction toward the future research of morphological analysis, it is better to distinguish implementation matters and knowledge description matters. In LAX, although the algorithm is the same in the implementation on PSI, a workstation, and personal computers, implementations are different from each other. This distinction may serve as a good example toward the actual matter of the development of natural language systems.

2. Japanese Dependency Analysis

2.1. Introduction

Traditionally, dependency analysis such as *kakari_uke* [Yoshida 72] and phrase structure analysis [Sells] tend to be discussed in different principles, so that analytical tools for dependency analysis usually have their base in modification deciding tables [Yoshida 72], and tools for phrase structure analysis had their base in parsers which accept type-2 syntax in the form of context free rules. [Kaplan & Bresnan]. Theoretically, the generation ability of dependency grammar is greater than that of context free grammars, and there might be a way to discuss both approaches on the same framework, but as a matter of fact, there was little work which strove to make a bridge between the two distinct approaches.

On the other hand, research activities related to sentence analysis tend to focus on disambiguation mainly using many kinds of heuristics. Such efforts chiefly tried to prioritize the results of the analysis and select one of the most plausible results [Fujisaki 89]. There was little effort to make machinery which enables us to limit the representatives step by step.

In this section, I will firstly illustrate a method which enables us to gradually reduce the number of results of sentence analysis using mathematical tools such as propositional logic problem solver [Sugimura 87a] [Sugimura 88a]. This kind of solver is becoming popular together with

the research activities on constraint programming [Dincbas] [Sakai], and Prolog II [Colmerauer].

Next to constraint analysis, a bridging method which enables us to write dependency relations in very simple declarative rules, and a process to carry out dependency analysis using a rule based parsing system will be presented [Sugimura 89f]. This is a result of our study on dependency grammar and my main contribution toward Japanese sentence analysis, in which I found that a class of dependency grammar can be written using the form of rewriting rules in context sensitive grammar (CSG for short).

2.2. Tools for Japanese Sentence Analysis

Before entering into the representation of Japanese sentence analysis, tools for such analysis will be presented. First is a solver for propositional constraints, and second is a parser for CFG and its extension to CSG.

2.2.1. Solver for Propositional Logic

Solver Using Lazy Evaluation in CIL

CIL, which is a language to describe sentence meanings and was developed at ICOT, facilitates a lazy evaluation mechanism and partially specified terms as its extension to Prolog [Mukai 88] [Sugimura 87b].

CIL = Prolog
 + Partially specified term
 + Lazy evaluation

The partially specified term is a data structure which can be regarded as a frame and represented as in the following example.

- (1) `term({a/X:(X?=Z),b/Y:(Y?=Z)}).`
- (2) `prompt> term(Z), Z!a = abc, Z!b = abc.`
 Yes.

Definition (1) is the specification of the data type and can be used in query (2). "a/X" means "X is the value of a slot 'a'". "?" displays the freeze mechanism attached to logical variable X. Therefore, $X = Y$ in (1) will not be evaluated until both X and Y are bound to some values. If this condition is not satisfied, unification $X=Y$ in (2) will fail when X and Y obtain their values. Query (2) utilizes the PST and unifies "abc" to each slot value. This freeze mechanism plays a crucial role in implementing the propositional solver in CIL which enables us to write logical constraints as the forms of CIL predicates as follows.

Prompt> constr(and(A,B), false), A = true.

A = true.

B = false.

Yes.

Prompt> constr(and(A,B), false).

A = Unbound.

B = Unbound.

Yes.

Implementation of "and" is as follows using the freeze mechanism. In the example, the predicate "bound" is a prolog predicate which becomes true when its argument is bound to ground atomic values.

constr(and(A,B), AB):-

freeze(A, (bound(B); freeze(B, (A=true,B=true, AB=true;AB=false)))),
freeze(B, (bound(A); freeze(A, (A=true,B=true, AB=true;AB=false)))).

A truth condition table realized in CIL is as follows.

$\wedge$	T	F	U
T	T	F	U
F	F	F	F
U	U	F	U

$\vee$	T	F	U
T	T	T	T
F	T	F	U
U	T	U	U

$\neg$	
T	F
F	T
U	U

$\Rightarrow$ is defined as $\neg A \vee B$ in CIL.

The weak negation $\sim$ can be utilized using CIL, and truth table is defined as follows.

$\sim$	
T	F
F	T
U	T

It is very important to note that in CIL, term rewriting is not carried out, therefore constraints like the following example can not be solved in CIL.

```
prompt> constr(and(A,B),AB), constr(and(A,C),AC),
        constr(or(AB,AC), true).
A = Unbound, B = Unbound, C = Unbound,
AB = Unbound, AC = Unbound.
```

In order to solve the constraints above, we should carry out term rewriting to get some canonical form for the propositional logic. In the above example, if we carry out normalization, we can easily rewrite constraints as follows.

$$(A \wedge B) \vee (A \wedge C) = A \wedge (B \vee C) = T$$

$$\Rightarrow A = T$$

It is well known that complexity for the propositional logic solver is NP complete, therefore we should meticulously utilize the full scale propositional logic solver. To utilize a full-fledged propositional solver, research on Graebner base propositional constraint solver will be one of the most efficient representatives[Sakai].

2.2.2. Logic Based Parser SAX

Basic Algorithm

SAX, which was originally developed by Matsumoto at ICOT, is a translator which inputs CFG or Definite Clause Grammar (DCG for short) [Pereira & Warren] rules and outputs Prolog clauses which can carry out parsing in a bottom-up breadth-first way [Matsumoto 86] [Matsumoto 87] [Matsumoto 88]. The word SAX also means this parsing program. SAX can output parallel programs which run on a parallel inference machine named PIM[Susaki]. We call SAX for parallel machines PAX.

Basically SAX can completely parse an input using context free rules. This is also able to parse DCG, though there are some limitations on writing grammar in DCG for SAX. Because SAX carries out parsing in a bottom-up breadth-first way, information which should be percolated from the top of parsed tree can not be passed through from top to bottom. After getting all the parsed trees, each interpretation can be unified with top-down information, but such kind of algorithm is not efficient.

Secondly, SAX utilizes a packed shared forest to represent partially parsed trees, so that the structure copy problem reduces the efficiency of parsing. There may be many heuristics which enable us to avoid this problem, but fundamentally, logical variables need to have some special treatment in the copy process.

In spite of these two limitations on grammar writing in DCG for SAX, SAX can carry out parsing at very high speed, because SAX pre-compiles its search space [Tanaka 79] and reduces pick up time for rule application of inactive edge [Sugimura 86c].

SAX transforms CFG rules into a Prolog program in the form of DCG. An example of the translation is given in Appendix C. As a facility to write a grammar and its semantics, SAX facilitates CIL as its semantic description language[Sugimura 86d].

SAX Extension to Type 0 Rules

While the DCG rule has only one symbol (type 2 grammar), SAX was extended to type 0 grammar, in that more than one symbol can be contained in the head. This extension is necessary for the description of dependency grammar, which will be explained later.

Generally, since syntactic analysis produces more than one interpretation, it is necessary to choose the best one. SAX provides the function for the preference calculation; it allows preference rule in {pref_rule}. In addition, the SAX system supports two types of extra conditions, one is executed in the parsing process, written as {extra}, the same as DCG's extra condition, and the other is executed after successful termination of parsing, written in &{delayed_extra}.

head ₁ , head ₂ , ..., head _m	(S1
→ body, {extra ₁ }:: {pref_rule ₁ },	(S2
.....	
body _n , {extra _n }:: {pref_rule _n }	(S3
& {delayed_extra},	(S4
&& {pref_rule}.	(S5

In the type 0 grammar, there is a risk that the rules will make an eternal loop, and currently in SAX, the user should verify whether there is possibility of an eternal loop or not.

2.3. Basic Principles of Dependency Analysis

As the basic principles for dependency analysis, I set two constraints. The first one is that no two modifying relations should cross each other as shown in Figure 17. I will refer this principle as UC1.

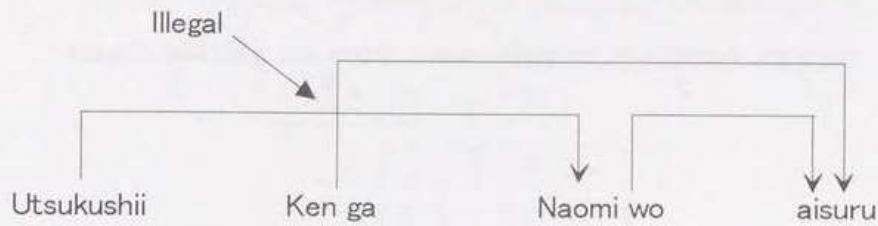


Figure 17 : An Example of Illegal Modifying Relations

The other principle is that all phrases except the last one should modify at least one phrase to the right of themselves. This principle will be called UC2.

2.4. Dependency Analysis Using Propositional Logic

To realize dependency analysis, the modification deciding table will be utilized[Sugimura 87e] as shown in Figure 18.

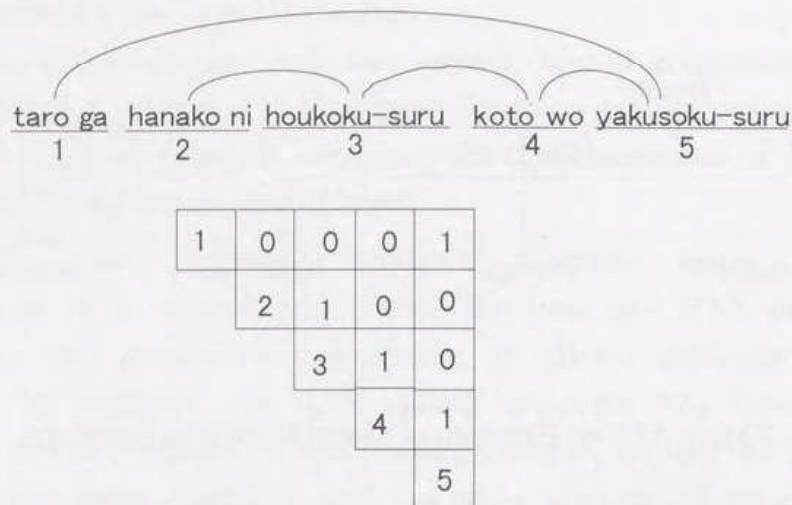


Figure 18 : An Example of a Modification Deciding Table

2.4.1. Basic Algorithm

In order to realize the UC1, constraints on the number of ${}_nC_4 = n!/(n-4)! \times 4!$ in which n is a number of phrases will be set first. For example, in the Figure 18, constraints will be set as follows.

taro ga hanako ni houkoku-suru koto wo yakusoku-suru
 1 2 3 4 5

1	U0	U1	U2	U3
	2	U4	U5	U6
		3	U7	U8
			4	U9
				5

$$(U1 \wedge (U5 \vee U6)) \vee (U2 \wedge (U6 \vee U8)) \vee (U5 \wedge (U1 \vee U8)) \vee (U6 \wedge (U1 \vee U2)) \vee (U8 \wedge (U2 \vee U5)) = F$$

Figure 19 : Constraints on a Modification Deciding Table

After setting the constraints, dependency analysis can be carried out.

2.4.2. Implementation

As the matter of fact, the number of constraints will be increased as shown in the following Table 7.

Table 7 : Number of Constraints corresponding to Length of Input Words

No of phrases	2	4	6	8	10	12	14	16	18	20
No of Constraints	0	1	15	70	210	495	1001	1820	3060	4845
No of phrases	22	24	26	28	30	32	34	36	38	40
No of Constraints	7315	10626	14950	20475	27405	35960	46376	58905	73815	91390

The number of constraints increase less than n^4 . Therefore, it is not beyond computational ability to set such kind of constraints before analysis. Actually, in CIL, or ESP which facilitates a lazy evaluation mechanism, we set the constraints for the sentences which have less than 40 phrases before the analysis. ¹

¹ Actually I asked the researchers who were in charge of freeze mechanism of PSI, some technical reasons blocked them to implement the compiled bind-hook which is freeze mechanism of PSI.

2.4.3. Analysis by Syntactic and Semantic Constraints

After applying the constraints UC1, and UC2, the other syntactic and semantic constraints are applied. Although there are many kinds of syntactic and semantic constraints in Japanese, crucial constraints will be shown here.

In the example shown in Figure 19, the following are examples of syntactic constraints.

SynC1) A phrase whose last word is post-position “ga”, “wo” or “ni” which are case markers, modifies only a predicative phrase.

SynC2) An adnominal phrase modifies only a nominal phrase. For example, the adnominal phrase “houkoku-suru” modifies only nominal phrase “koto wo”.

After checking the syntactic constraints, the modification deciding table in Figure 19 looks like that in Figure 20.

taro ga hanako ni houkoku-suru koto wo yakusoku-suru
1 2 3 4 5

1	0	U1	0	U3
	2	U4	0	U6
		3	1	0
			4	1
				5

$(U1 \wedge U6)=F$

Figure 20 : Resultant Constraints on a Modification Deciding Table

The semantic constraints on dependency relations correspond to a consistency check of semantic features in ordinary natural language processing systems. The following is an example of a semantic constraint.

SemC1 A non-animate nominative phrase cannot modify an emotional predicative phrase.

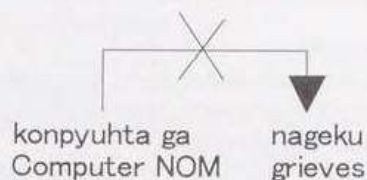


Figure 21 : An Example of Semantic Constraints

Semantic analysis is carried out together with syntactic analysis. If the syntactic rules of modifying relations allow modifying relations between phrases to become 1, logical variables in the dependency deciding table remain unbound and the relationship between the semantic descriptions of phrases is checked. This check is based on SemC1. For example, in Figure 22 the semantic checking mechanism is used to reduce ambiguity in U6. In this case, modifying relations cannot be reduced because the semantic features in the phrases “Hanako ni” and “yakusoku-suru” can be unified.

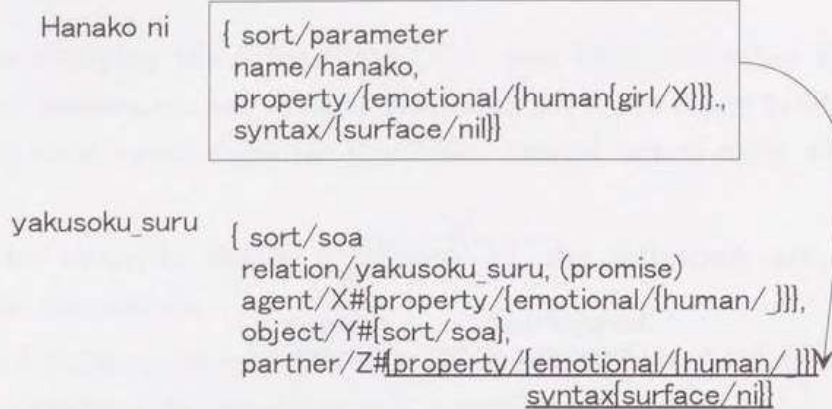


Figure 22 : An Example of Semantic Check on Modifying Relation

To check semantics, the properties of each phrase are unified. In this example, because ambiguities in the modifying relation cannot be reduced, the result of analysis is passed to contextual analysis in which constraints on the modification deciding table are propagated with partial interpretation like that in Figure 23.

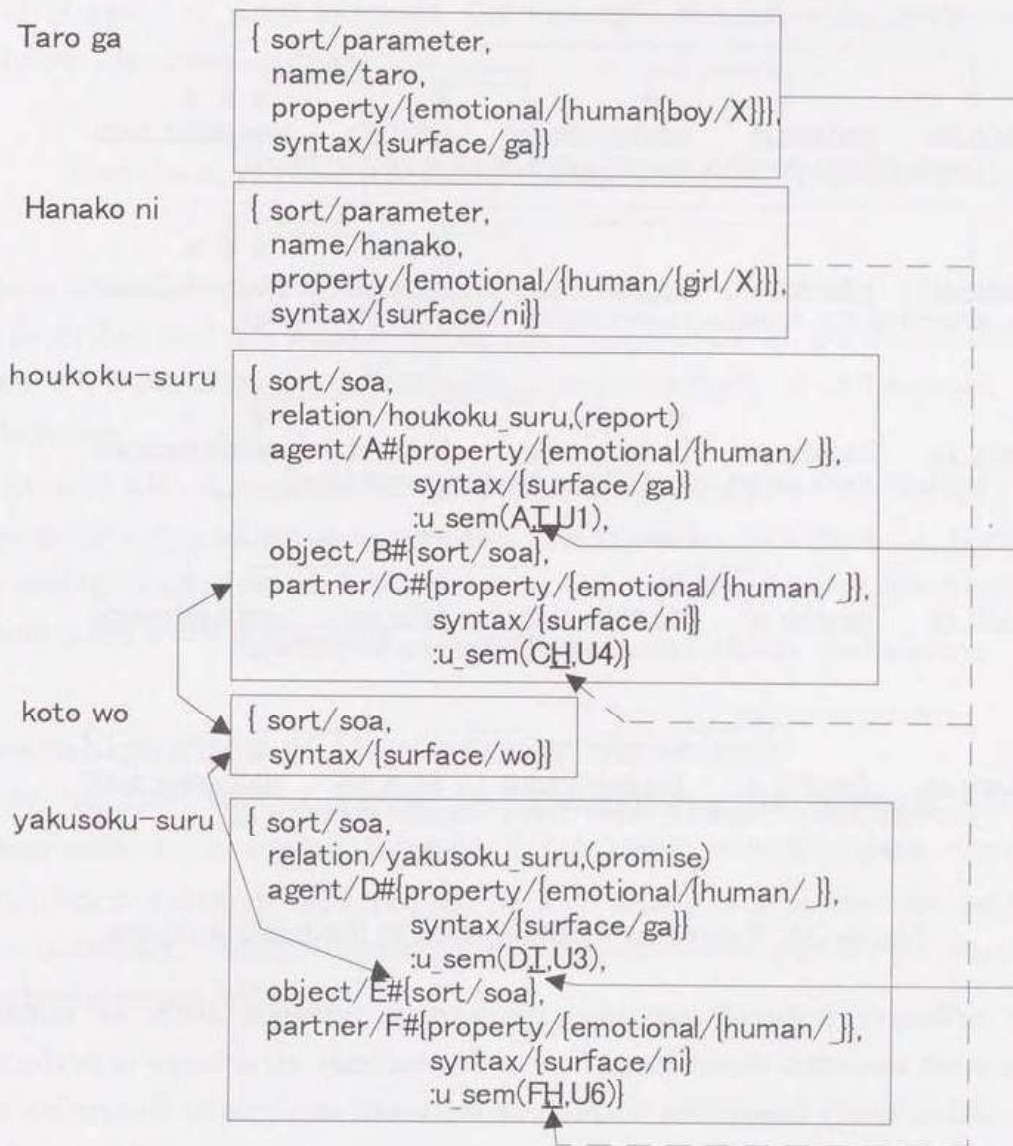


Figure 23 : An Example of Ambiguous Semantic Relations

The predicate “u_sem” is defined as follows.

```
u_sem(A,B,Sign):- freeze(A,(var(F),F=1,(A=B,Sign=1;true))),
                  freeze(Sign,(var(F),F=1,(Sign=1, A=B;true))).
```

Ambiguities in Figure 23 correspond to the following five dependency structures as shown in Figure 24.

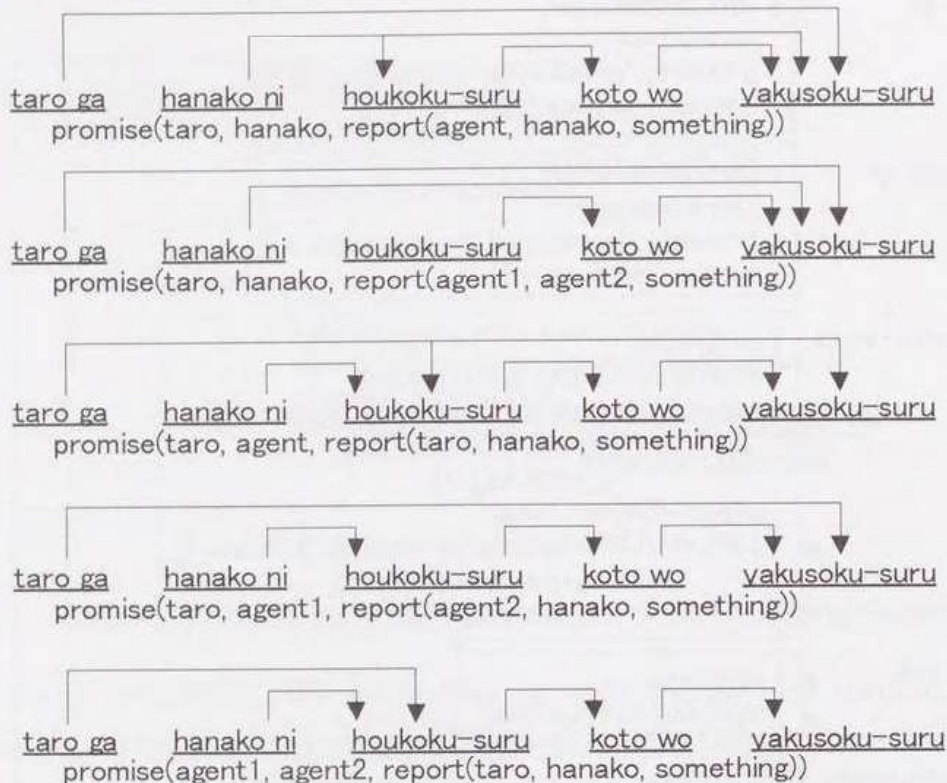


Figure 24 : Examples of Ambiguities in Syntactic Analysis

In ordinary natural language processing systems, such as machine translation systems, disambiguation of dependency structures is performed by applying many heuristics within one sentence analysis to determine only one interpretation. This often generates wrong interpretations. Disambiguation should be done using contextual information in contextual analysis, and syntax and semantic analysis modules should pass feasible ambiguity structures to contextual analysis. In the next section, a basic method to bridge sentence disambiguation and context information will be studied.

2.4.4. Contextual Analysis of a Sentence by Logical Constraints

Before entering into contextual analysis, the terminology for ambiguity and vagueness must be defined. Japanese sentences in most cases have ellipsis in which a phrase indicating the agent or object can be omitted when it is clear

what is agent or what is object. For example, the following sentence is a complete Japanese sentence.

Hanako ni yakusoku-shita. (Someone promised Hanako something)

In this case, the agent and object are vague. Vague means that something is not described and it is hard to list up the representatives. On the other hand, when we have some list of representatives for lack of information, it is ambiguous.

Even if all ambiguities in the dependency structures are solved, some vagueness may remain in semantics. For example, in Figure 24, there are five ambiguities and each interpretation has vagueness which is represented by using the word "something", or "agent".

Disambiguation of Dependency Structure

As we saw in the Figure 20, ambiguity in an example sentence corresponds to a logical formula $U1 \wedge U6 = F$, which means that the dependency relation represented in $U1$ and $U6$ will not be satisfied simultaneously. Each dependency relation corresponds to semantic interpretation as follows.

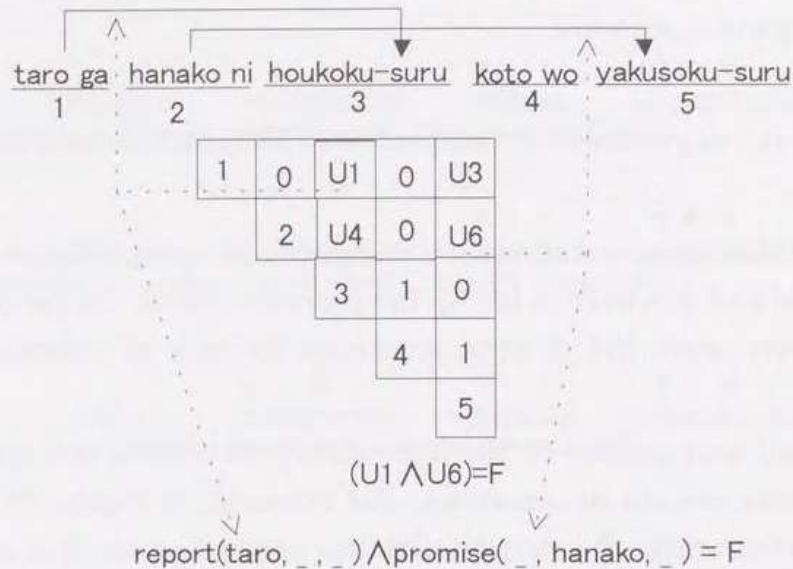


Figure 25 : An Example of Syntactic Constraints between Two Interpretations

Therefore, in order to disambiguate the sentence in the example, the main task to be carried out in the context analysis is to decide whether "Taro" can be an agent of the predicate report, or "Hanako" can be a recipient of the predicate "promise". In such a case, if we can get inference from the context that "Taro" cannot be an agent of report, we can easily decide the syntactic structure of an example sentence. Although we have such a possibility to disambiguate sentence interpretations, it is still hard to disambiguate the sentence because there is usually little information which strongly denies some interpretation. Interpretations of a sentence usually carry partial information which is supported by much contextual information, and such a mechanism gives a sentence efficiency.

To solve the problem which is caused by partiality of a sentence interpretation, there are many kind of methods which will be discussed in a later section.

2.5. Dependency Analysis Using Type 0 Rules

2.5.1. Basic Algorithm

This section introduces a set of logic grammar rules, with which we can get dependency structures from an input sentence. Dependency structures are constructed from constituents of a sentence, and modifying relations between them. In the previous section, dependency relations can be defined in a declarative way, but dependency relations remain to be represented in a modifying table with which we can think of the relations in procedural manners. With the rules presented here, we can write also the dependency constraints in declarative rules.

I will show that modifying relations in dependency structures can be written as a set of simple rewriting rules including type-0 and type-1 rules. As a matter of fact, termination of the rewriting process with the proposed rules is guaranteed.

It is well known that there is a syntactic and semantic idiosyncratic gap between Japanese and English. Japanese has dependency structure in which one phrase modifies other phrases. On the other hand English has phrase structure.

The idiosyncratic gap between dependency structures and phrase structures reflects not only syntactic idiosyncrasy but also semantic idiosyncrasy.

As for rewriting rules, we cannot write context free rules for dependency structures in which one phrase modifies more than two phrases. Machine translation systems usually utilize procedures in order to get dependency structures of agglutinative languages such as Japanese or Korean.

As the main feature of our system, we can analyze the modifying relations in dependency structures in declarative rules with graphs which correspond to semantic structures.

For disambiguation of the result of sentence analysis, I propose a constraint approach. In our approach, logical constraints between results of parsing can be propagated into context analysis.

First, we will summarize dependency grammars. Then, simple rules for dependency structure will be introduced.

2.5.2. Grammar Rules for Dependency Structure

Now a tiny set of dependency grammar is introduced.

The following rules correspond to the UC1, and UC2. We will use the symbol “word” to note Japanese word with case marker.

<code>word₁, arc, word₂</code>	$\rightarrow$	<code>word₁, word₂.</code>	(Rr1)
<code>word₁, arc, word₂</code>	$\rightarrow$	<code>word₁, (arc; []), word_arc_array, word₂.</code>	(Rr2)
<code>word_arc_array</code>	$\rightarrow$	<code>word, arc.</code>	(Rr3)
<code>word_arc_array</code>	$\rightarrow$	<code>word, arc, word_arc_array.</code>	(Rr4)
<code>sentence</code>	$\rightarrow$	<code>word_arc_array, word_tail.</code>	(Rr5)

Rr1 is fundamental to display the modifying relation between adjacent phrases as follows. After deciding the modifying relation between adjacent words, the object string to rewrite became “phrase arc phrase”.



Rr2) is a rule which realizes the modifying relation between two phrases which are apart from each other as follows. This rule also enables us to realize UR1 in which no two modifying relations can cross each other. After applying this rewriting rule, the string to rewrite becomes simplified to a modifying relation.

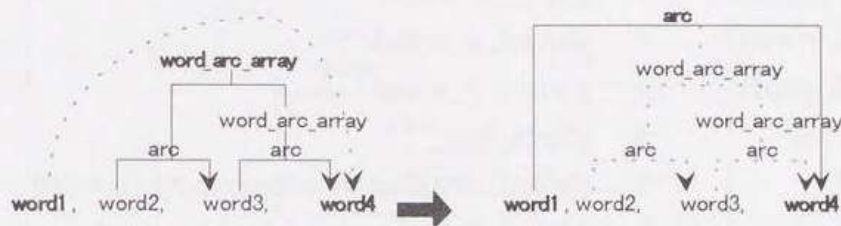


Figure 26 : An Example of Dependency Structure realized by Rewriting Rules.

Rr3) and Rr4) represent a sequence of modifying relations and also eliminate a word which has no relation to other phrase.

Rr5) defines a sentence which is a sequence of modifying relations.

Rewriting rules from Rr1) through Rr5) realize a set of rules to recognize a sequence of words as a modifying relation, and termination of rewriting can be guaranteed because there is no critical pair which generates cyclic rewriting. Every sequence of phrases are reduced to `ph_arc_array`, and this `ph_arc_array` will not generate longer strings compared to `ph_arc_array`.

2.5.3. Implementation

Fundamentally, a set of rules introduced in Rr1) through Rr5) is enough to accept a sequence of phrases as modifying relations, but in Japanese phrases, there are some phenomena which makes sentence analysis difficult. One of the most famous is coordination of phrases.

In order to realize effective implementation, we should study about coordination in the style of modifying rewriting rules.

word, arc, word	→	word, word.	(Rr1
word, arc, word	→	word, word_arc_array, word.	(Rr2 ₁
word, arc, ph	→	word, arc, word_arc_array word.	(Rr2 ₂
word_arc_array	→	word, bar.	(Rr3
word_arc_array	→	word, bar, word_arc_array.	(Rr4
sentence	→	word_arc_array, word_tail.	(Rr5
begin, gapH	→	[begin], g_word.	(Rr6
begin, pword	→	[begin], g_word.	(Rr7
word, pword	→	gapH, g_word.	(Rr8
word, gapH	→	gapH, g_word.	(Rr9
pword, pword	→	pword, g_word.	(Rr10
pword, gapH	→	pword, g_word.	(Rr11
word, bar	→	gapH, bar.	(Rr12
pword	→	pword, arc, (word_arc_array;[]), word.	(Rr13
pword	→	pword, (word_arc_array;[]), word.	(Rr14
word	→	pword.	(Rr15

In the rules above, every pre-terminal name is "g_word". First of all, g_words are subject to rewriting rules, from Rr6) through Rr15), and words which may have a coordinating word will have a non-terminal symbol name "pword". "pword" will be make a special combination with its counterpart, and become a word which is a subject to a modifying relation. These procedures are carried out as shown in the following example.

Finally, I reached the point of defining formally what we have been doing in the constraint approach toward discourse analysis.

In this section, I will develop in depth my theory which successfully pursues the fundamental relations between discourse analysis and sentence analysis[Sugimura 88c][Sugimura 89g]. In the presentation, I will utilize DCG grammar as a formal device which could be enlarged to the framework of formal dependency analysis which was presented in section 2.4..

2.6.1. Logic Grammar

In a logic grammar like DCG, analysis can be considered the process to prove the predicate "sentence" with given inference rules called a grammar and an input sentence. Partial trees which are obtained in the process of the analysis can be considered as a theorem which is derived from the input sentence and the grammar rules.

First, the inference rules as grammar rules can be defined as follows.

$$h(Sh, A, An) \leftarrow b1(S1, A, A1) \wedge \dots \wedge bn(Sn, An-1, An) \wedge \quad (1)$$

$$rel(Sh, (S1, \dots, Sn)) \quad (2)$$

$$bj(Sj, A, B) \leftarrow c(A, B) \wedge B = A + 1. \quad (3)$$

Formula (1) and (2) are the inference rules given as a grammar. Logical variables $Sh, Sj(1 \leq j \leq n)$ make logical predicates h and $bj(1 \leq j \leq n)$ true. Logical variables $A, Aj(1 \leq j \leq n)$ indicate a specific sub-string of the input sentence to make predicate h , and bj true, where each word of the input sentence is numbered from the left hand side word. For example, $h(Sh, A, An)$ indicates that a sub-string from the word numbered A through An can be interpreted as "h".

The predicate at (2) holds when there holds some relation among variables from $S1$ to Sn .

Predicate $bj(Sj, A, B)$ at (3) is a predicate given as a dictionary entry which holds when there is a word "c" at the position A .

2.6.2. Multiple Interpretations and Context

Let us consider sentence analysis to prove the predicate “sentence(Sem, 1, L+1)” from the given inference rules, dictionary rules and an input sentence. Generally, more than one “sentence” can be proved corresponding to one input sentence. In order to handle multiple interpretations, a new predicate which is underlined is introduced to the previous grammar as follows.

$$\begin{aligned} & \text{Sentence}(\text{Sem}, A, A_n) \wedge \underline{\text{Sem}} \in I \\ & \leftarrow b_1(S_1, A, A_1) \wedge \dots \wedge b_n(S_n, A_{n-1}, A_n) \wedge \\ & \quad \text{rel}(\text{Sem}, (S_1, \dots, S_n)) \end{aligned}$$

“I” is a set with semantic information which makes predicate “Sentence” true.

Next, semantic information will be expanded to handle discourse information as, even in a sentence, discourse varies [SUGIMURA 86]. Therefore, semantic information will be paired with discourse parameters one by one. For instance, if discourse varies according to the surface structure, a grammar rule corresponding to the discourse is presented as follows.

$$\begin{aligned} & h((DS_b, S_b), A, A_2) \\ & \leftarrow b_1((DS_a, S_1), A, A_1) \wedge b_2((DS_b, S_2), A_1, A_2) \wedge \\ & \quad \text{rel}((DS_b, S_b), ((DS_a, S_1), (DS_b, S_2))) \end{aligned}$$

In the example above, the discourse situation was changed from DS_a to DS_b , between predicate b_1 and b_2 , and last, the discourse situation becomes DS_b .

In the dictionary rule, in the discourse, there is a word c , and from the situation, the meaning of “ c ” can be derived, therefore words in the dictionary can be defined as follows.

$$\begin{aligned} & h((DS, S_j), A, B) \wedge \text{rel}((DS, S_j), (DU, c)) \wedge DU \models c \\ & \leftarrow c(A, B) \wedge B = A + 1 \end{aligned}$$

c is dependent on the discourse situation so that $DU \models c$ holds.

2.6.3. Constraints on Discourse

Sentence analysis which was carried out in the rules above gives rise to the following constraint.

$$\begin{array}{l} \text{Axiom D1:} \\ \exists DS \models S \text{ where } (DS, S) \in I \end{array}$$

This constraint shows that a sentence should have at least one interpretation in its discourse. This constraint can be applied to the case when a set I has two elements as shown in the example below.

$$(DS \models S_1) \vee (DS \models S_2)$$

Next to the Axiom D1, I will show an important axiom which also plays an important role in the discourse.

$$\begin{array}{l} \text{Axiom D2} \\ (DS \models S_h) \rightarrow (DS_1 \models S_1) \wedge \dots \wedge (DS_n \models S_n) \\ \text{where } \text{rel}(S_h, ((DS_1, S_1), \dots, (DS_n, S_n))) \end{array}$$

The axiom D2 declares that if there holds an interpretation in some discourse situation DS which is constructed from its parts, the parts also hold in each discourse situation.

2.6.4. Sentence Analysis

First of all, sentence analysis should be done without duplicating the sub-tree, because we would like to avoid making trivial ambiguities of the sentence interpretation which are caused by the duplicated predicates. Therefore, a set ST with partial trees (theorems) as its elements is defined as follows.

$$\begin{array}{l} ST = \{St_i \mid St_i \text{ is a sub-tree with index } i\} \\ \text{where } \forall i \forall j (St_i = St_j \rightarrow i = j) \end{array}$$

This condition is realized with parser SAX, or BUP [Matsumoto], or LangLab [Konno].

2.6.5. Constraint Analysis in Discourse

Under the constraints shown in the previous discussion, I will present a method to analyze an English sentence basically by phrase structure grammar, and a Japanese sentence basically by dependency grammar.

The crucial point of this section is that we have finally reached the point to say that we can analyze a sentence completely logically in discourse. Therefore, this section can be considered as the pinnacle of this dissertation.

2.6.6. Phrase Structure Analysis in Discourse

Let us consider the sentence "You saw John on that", which has four interpretations as follows. In the example, I utilize the semantic representation form of Situation Semantics in which an attribute and its value are expressed in the form "attribute/value". Curly braces express a set of these pairs which can be regarded as frame or partial information.

```
{relation/see, tense/past, agent/you, object/{parameter/John}, location/that}  
{relation/see, tense/past, agent/you, object/{parameter/John}, location/that}}  
{relation/saw, tense/present, agent/you, object/{parameter/John}, location/that}  
{relation/saw, tense/present, agent/you, object/{parameter/John}, location/that}}
```

In the sentence analysis, we get the results as follows.

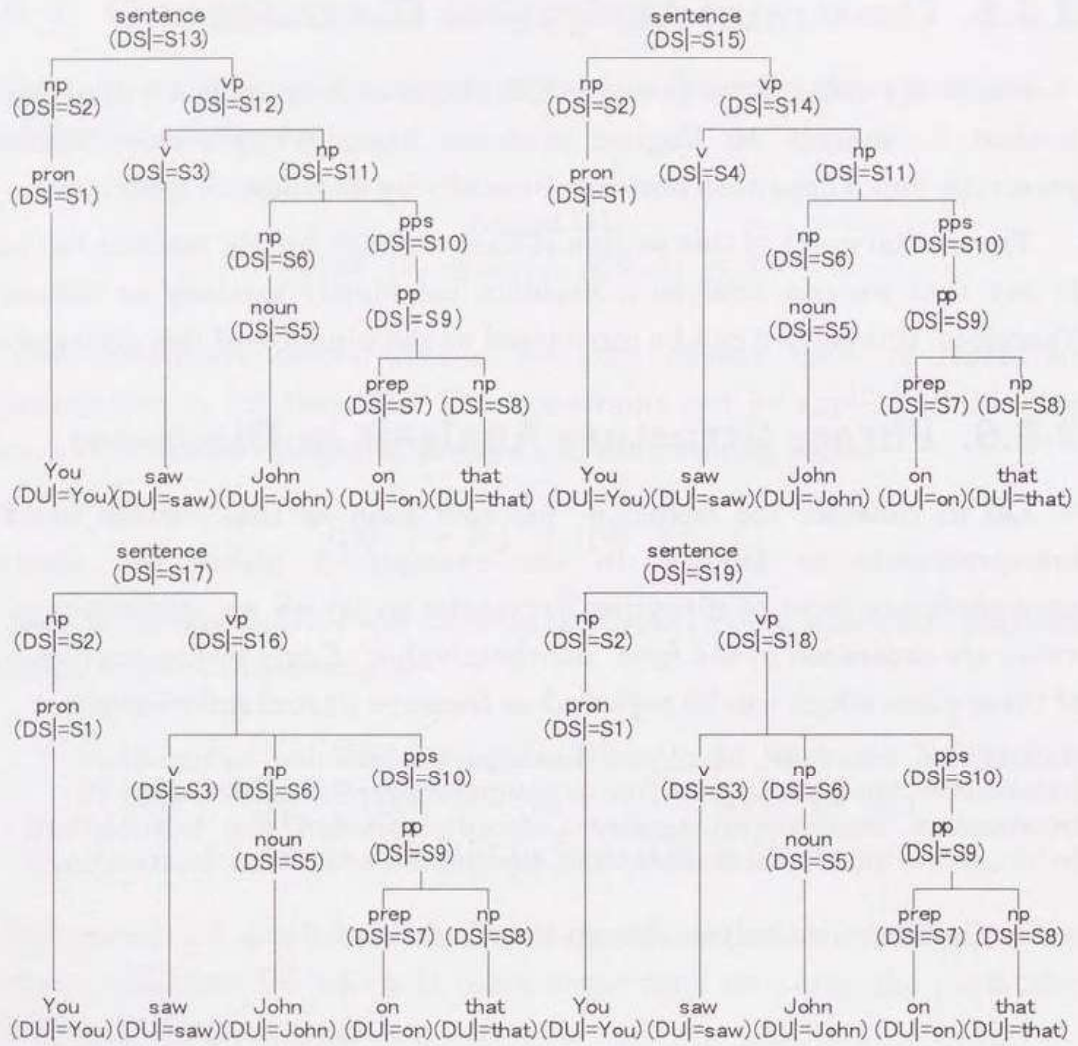


Figure 28 : An Example of Ambiguities of Sentence Analysis

In these sub-trees, we can get logical consequences under the axiom D1, and D2 as follows.

From Axiom D1:

$$(DS \mid = S13) \vee (DS \mid = S15) \vee (DS \mid = S17) \vee (DS \mid = S19) = T$$

From Axiom D2

$$(DS \mid = S13) \rightarrow (DS \mid = S2) \wedge (DS \mid = S12)$$

$$(DS \mid = S2) \rightarrow (DS \mid = S1)$$

$$(DS \mid = S12) \rightarrow (DS \mid = S3) \wedge (DS \mid = S11)$$

$$\begin{aligned}
(DS \models S11) &\rightarrow (DS \models S6) \wedge (DS \models S10) \\
(DS \models S6) &\rightarrow (DS \models S5) \\
(DS \models S10) &\rightarrow (DS \models S9) \\
(DS \models S9) &\rightarrow (DS \models S7) \wedge (DS \models S8) \\
(DS \models S15) &\rightarrow (DS \models S2) \wedge (DS \models S14) \\
(DS \models S14) &\rightarrow (DS \models S4) \wedge (DS \models S11) \\
(DS \models S17) &\rightarrow (DS \models S2) \wedge (DS \models S16) \\
(DS \models S16) &\rightarrow (DS \models S3) \wedge (DS \models S6) \wedge (DS \models S10) \\
(DS \models S19) &\rightarrow (DS \models S2) \wedge (DS \models S18) \\
(DS \models S18) &\rightarrow (DS \models S4) \wedge (DS \models S6) \wedge (DS \models S10)
\end{aligned}$$

From the utterance:

$$\begin{aligned}
(DS \models S1) &\rightarrow (DU \models \text{You}) \\
(DS \models S3) &\rightarrow (DU \models \text{saw}) \\
(DS \models S4) &\rightarrow (DU \models \text{saw}) \\
(DS \models S5) &\rightarrow (DU \models \text{John}) \\
(DS \models S7) &\rightarrow (DU \models \text{on}) \\
(DS \models S8) &\rightarrow (DU \models \text{that}) \\
DU \models \text{You} \quad DU \models \text{saw} \quad DU \models \text{Jon} \quad DU \models \text{on} \quad DU \models \text{that}
\end{aligned}$$

On these logical relations, we can carry out various kind of formal analysis of the sentence in discourse using some kinds of propositional logic problem solvers.

Problem Analysis with Passive Solver with Freeze

One of the easiest ways to solve the logical conditions is to utilize the propositional logic solver with freeze or lazy evaluation which was introduced in section 2.2.2..

In the method to utilize the solver with freeze mechanism, no term is rewritten to canonical form, so that we cannot get any logical consequences from the theorems above.

However, if we can get some additional predicate which holds, we can reduce the ambiguity of the sentence interpretations. For instance, in the case that $(DS \models S3)$ which means "saw is to see in past tense", is negated,

ambiguities will be reduced from 4 to 2. This problem-solving step was carried out as follows.

$$\begin{aligned}
& (DS \mid \neq S3) \quad \wedge ((DS \mid = S12) \rightarrow (DS \mid = S3) \wedge (DS \mid = S11)) \\
& \quad \wedge ((DS \mid = S13) \rightarrow (DS \mid = S2) \wedge (DS \mid = S12)) \\
& \quad \wedge ((DS \mid = S16) \rightarrow (DS \mid = S3) \wedge (DS \mid = S6) \wedge (DS \mid = S10)) \\
& \quad \wedge ((DS \mid = S17) \rightarrow (DS \mid = S2) \wedge (DS \mid = S16)) \\
& \text{then, } (DS \mid \neq S12) \wedge (DS \mid \neq S13) \wedge (DS \mid \neq S16) \wedge (DS \mid \neq S17) \quad (LC1)
\end{aligned}$$

After getting the logical consequences in (LC1, if we get a new predicate $(DS \mid \neq S11)$ that means "John was not on that", logical consequences can be derived as follows.

$$\begin{aligned}
& (DS \mid \neq S11) \quad \wedge ((DS \mid = S14) \rightarrow (DS \mid = S4) \wedge (DS \mid = S11)) \\
& \quad \wedge ((DS \mid = S15) \rightarrow (DS \mid = S2) \wedge (DS \mid = S14)) \\
& \text{then, } (DS \mid \neq S15) \\
& \text{then, } (DS \mid \neq S13) \wedge (DS \mid \neq S15) \wedge (DS \mid \neq S17) \\
& \text{then } (DS \mid = S19)
\end{aligned}$$

As shown in the example above, we can get a solution in a logical way.

The disadvantage of this method is the inability to derive some predicate which holds as a logical consequences. In the next session, some active approach toward the problem solver will be discussed.

2.6.7. Dependency Structure Analysis in Discourse

Let us consider the example in Figure 24. In the example, modifying relations which are displayed with thick lines are common among all the interpretations. These common modifying relations correspond to the interpretations as follows

$$\text{promise}(X, Y, \text{report}(Z, A, B))$$

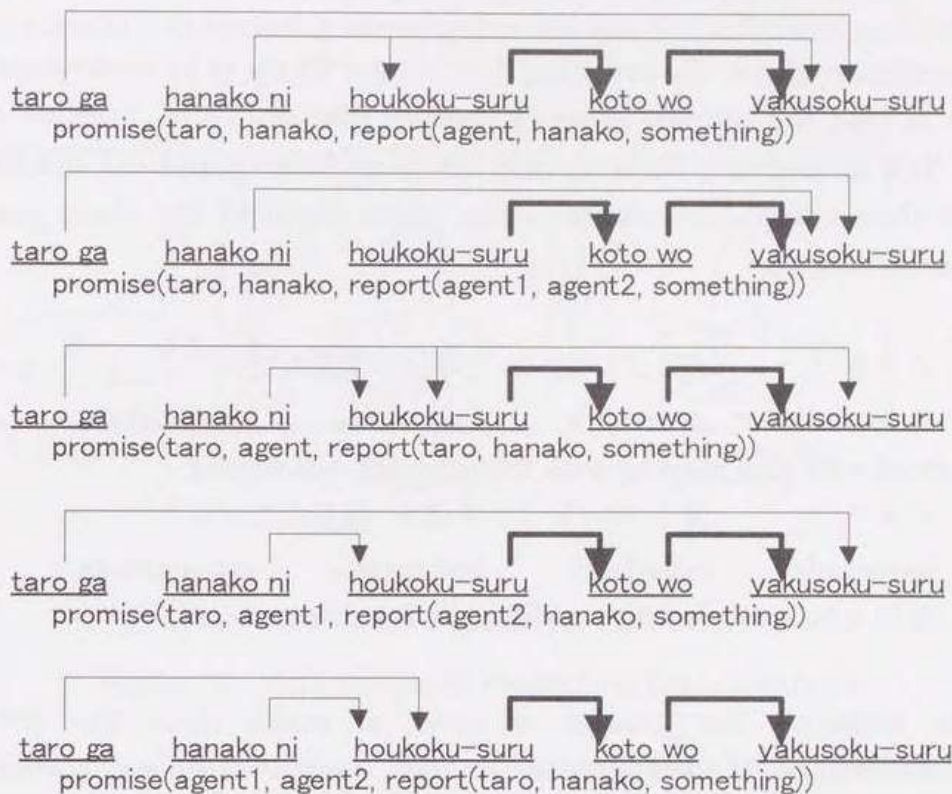


Figure 29 : An Example of Valid Interpretations in Ambiguous Sentence Analysis

Although there are many variants in the interpretations, the common interpretations become true as the logical consequence of D1, and D2.

2.7. Related Methods

2.7.1. Dependency Analysis Using Modification Deciding Network

Concerning dependency analysis for a machine translation system, we tried to develop another kind of analysis engine using “neural network technology” [Ishikawa 92]. We call our method “Modification Deciding Network” (MDN for short). The MDN has available a number of heuristic rules which give it clues as to the most likely meaning relations between the words of a sentence. For example, let us consider the sentence below.

They filmed the child playing with the camera.

In the sentence above, the meaning that is most likely to be understood by a human is that the camera is being used to film the child, because of the strong link in meaning between the action of filming and the instrument used in that action; the camera. Thus, MDN captures the above example sentence as follows:

この カメラで 遊んでいる 子供を 撮影した.

Kono kamerade asondeiru kodomowo satsueishita.

1) The child who was playing with this camera was filmed.

この カメラで、 遊んでいる 子供を 撮影した.

Kono kamerade asondeiru kodomowo satsueishita.

2) The child who was playing was filmed with this camera.

In this example, the passive is used to make clear the different interpretations of the ambiguity. In the second sentence, there is a punctuation character which the MDN uses to resolve the ambiguity and so produce the correct interpretation. The uses of the punctuation character are not syntactically determined like the comma in English and its use depends more on meaning relations. The effect on meaning that the use of punctuation characters has is contained as one of the heuristic rules which the MDN uses. Thus, while the first sentence has two interpretations, the second has only one.

2.8. Comparison, Evaluation and Discussion

In order to examine the idea in which we postpone the sentence disambiguation process to the appropriate time which there will be sufficient information to resolve those ambiguities we applied the newly emerging paradigm of constraint logic programming into Japanese sentence analysis based on dependency relations. At the beginning of our approach to the sentence analysis, we thought that constraint logic programming was only a

useful device to handle a set of ambiguous meanings. This is because we took the constraint programming paradigm to handle the extension as follows.

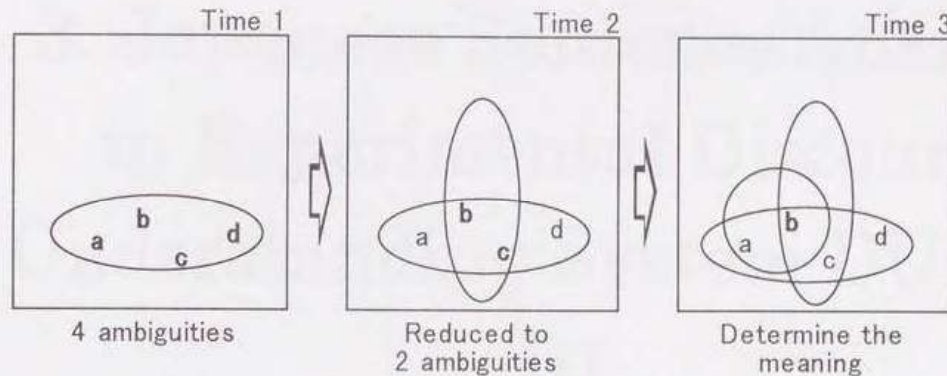


Figure 30 : An Example of Procedural Disambiguation

In Figure 30, there are four ambiguities (a,b,c and d) for the sentence interpretation at time 1. After a while at time 2, we got a new constraint which eliminates the interpretation a and d. And finally at time 3, we got another constraint which is displayed by a circle, and determined the meaning to be "b". This is a first idea to utilize the constraint logic programming paradigm.

Although the example seems to enable us to reduce the ambiguities of a sentence step by step, there should be a discussion that such kind of process is bound to a procedural model and limits the ability of this sophisticated mathematical device. Basically, the constraint logic programming paradigm hates to be stuck to the procedural model. It is suitable for constraint programming to utilize the declarative model.

On the other hand, there are alternatives to consider the sentence analysis as a procedure to get an extension of interpretations of a sentence, or as a procedure to get the intention of interpretations of a sentence. The approach based on intention is effective when we handle the vagueness in the analysis because we can not count up all the constituents of the interpretation.

Figure 1. A schematic diagram of the experimental setup. The diagram shows a cross-section of a container with a liquid surface. A light source is positioned above the surface, and a camera is positioned below it. The light source emits a beam of light that passes through the liquid surface and is reflected back to the camera. The camera is positioned at a distance z from the surface, and the light source is positioned at a distance z_0 from the surface. The diagram also shows the coordinates x and y of the light source and camera.



Figure 1. A schematic diagram of the experimental setup. The diagram shows a cross-section of a container with a liquid surface. A light source is positioned above the surface, and a camera is positioned below it. The light source emits a beam of light that passes through the liquid surface and is reflected back to the camera. The camera is positioned at a distance z from the surface, and the light source is positioned at a distance z_0 from the surface. The diagram also shows the coordinates x and y of the light source and camera.

The experimental setup is shown in Figure 1. A light source is positioned above the surface of a liquid, and a camera is positioned below it. The light source emits a beam of light that passes through the liquid surface and is reflected back to the camera. The camera is positioned at a distance z from the surface, and the light source is positioned at a distance z_0 from the surface. The diagram also shows the coordinates x and y of the light source and camera. The light source is positioned at a distance z_0 from the surface, and the camera is positioned at a distance z from the surface. The light source emits a beam of light that passes through the liquid surface and is reflected back to the camera. The camera is positioned at a distance z from the surface, and the light source is positioned at a distance z_0 from the surface. The diagram also shows the coordinates x and y of the light source and camera.

The experimental setup is shown in Figure 1. A light source is positioned above the surface of a liquid, and a camera is positioned below it. The light source emits a beam of light that passes through the liquid surface and is reflected back to the camera. The camera is positioned at a distance z from the surface, and the light source is positioned at a distance z_0 from the surface. The diagram also shows the coordinates x and y of the light source and camera. The light source is positioned at a distance z_0 from the surface, and the camera is positioned at a distance z from the surface. The light source emits a beam of light that passes through the liquid surface and is reflected back to the camera. The camera is positioned at a distance z from the surface, and the light source is positioned at a distance z_0 from the surface. The diagram also shows the coordinates x and y of the light source and camera.

3. Japanese Sentence Analysis in Experimental Discourse Understanding System DUALS- III

3.1. Introduction

At ICOT, the research aim was focused on discourse understanding [Yokoi 86]. To tackle this difficult theme, I and my colleagues at ICOT had to create a vast amount of experimental devices which would support our research activities at first.

The first matter in the discussion on our research direction was how to develop dictionaries. The discussion finally concluded in the creation of a new laboratory EDR.

The next matter to discuss was the research principle. At that time, there were many works related to unification grammar formalism such as LFG [Kaplan & Bresnan], GPSG [Gazdar], HPSG [Pollard], and so on. Research working groups were launched to study these frameworks at ICOT. At the same time, we encountered a new theory on discourse, which is the theory of situation semantics [Barwise].

Situation semantics gave us a fairly good principle not because it was simply new, but because it seemed to give us a theoretical foundation to manipulate

the complicated phenomena of discourse. We at ICOT devoted ourselves to tackling a new framework to represent the many relations between discourse and sentence. On the other hand, we were engaged to develop suitable tools to handle discourse in logic programming [Sugimura 89d]. These two activities are unified and displayed through the Discourse Understanding Aimed at Logic based System (DUALS for short) [Sugimura 89b][Sugimura 89c].

In this section, I will present Language Tool Box (LTB for short) which is a set of fundamental tools to handle discourse information [Sugimura 88e]. Secondly, I will present each tool [Sugimura 88e]. My major contribution to LTB was to design each tool as the leader of LTB project.

3.2. Language Tool Box LTB

3.2.1. Introduction

A natural language understanding system must meet many kinds of demands.

As the interface system between computer systems and their users, natural language systems tend to be expected to be able to understand not only what users say but also what they intend. A natural language system is expected to make the interface more comfortable.

A natural language system for researchers of linguistics or computer linguistics should be very flexible. It should enable researchers to input their ideas freely, such as grammars and lexical entries, so that it can be used to verify their research ideas. There should also be some debugging or tracing tools to check how a grammar or a lexicon works.

A natural language system for software system development should have modularity, high performance, and compatibility with other application software such as an expert system. Maintenance tools should be prepared in the natural language system. In this regard, I think it is important to develop glueing tools which support us in constructing a system from functional modules like a morphological analyzer, or a parser.

The Language Tool Box (LTB for short)[Sugimura 89d] has been developed to meet these demands.

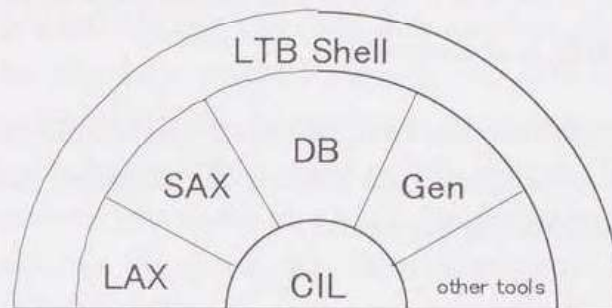
Features of the LTB include the following.

- It is intended to provide up-to-date logic based tools for natural language understanding research.
- It accommodates many types of software environments such as debuggers and editors.
- All the tools are written in ESP so that they have high modularity and compatibility with each other.

We believed that a variety of natural language systems could be developed based on LTB.

3.2.2. Structure of LTB

As shown in Figure 31, the LTB was constructed from the following component tools. Some of them have already been introduced.



Configuration of the LTB

Figure 31 : Configuration of the LTB

Shell: The LTB shell [Takizuka 88b] facilitated the interaction with tools with the LTB. The shell itself could also be executed in a parallel logic programming language.

Database: The collection of Japanese lexicon, Japanese morphological grammars and syntactic grammars [Sano 88][Takizuka 88a].

CIL: The basic programming language of the LTB. Every tool in the LTB had access to CIL and its software environment. Details have already been introduced in the section 2.2.1.

LAX : This tool has already been introduced in section 1

SAX : This tool has already been introduced in section 2.2.2.

GEN: The Japanese sentence generator, whose input is a frame structure written in CIL, outputs Japanese surface sentences. It is beyond the scope of this thesis to describe the generation part.

3.2.3. LTB Shell

The LTB shell has the following features.

- It managed all the communications in the LTB. It controlled all the information flow among the user and application tools in the LTB.
- It accommodated the notify function which informed the shell of data communication among processes, so that the shell could measure the execution interval of a process in the LTB.
- It controlled all the processes in the LTB.
- It provided a standard window. The window had a multi-lingual menu so that users could use LTB tools in either English or Japanese.
- It provided help instructions.

On parallel machines like PIM, the shell will enable each process to move simultaneously. Communications among the parallel processes in the LTB are performed as shown in Figure 32.

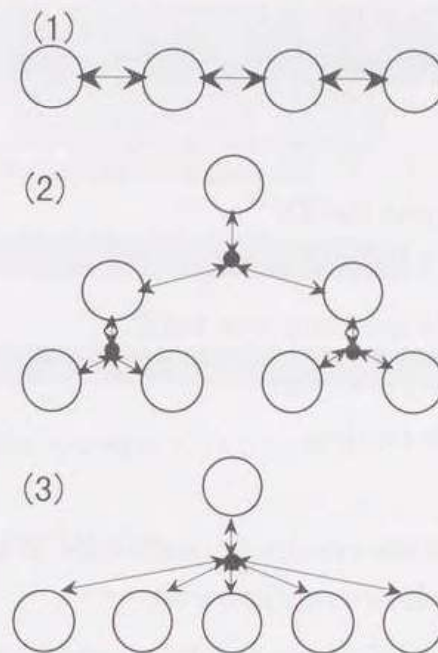


Figure 32 : Data Flow Model for LTB Shell

In Figure 32, graph(1) shows the data flow through a pipe-line in which there is no need for the data sender to specify the data receiver. Graph (2) enables communication between hierarchical processes in which the data sender should specify the direction, right or left. Graph (3) enables network communication between processes in which the data sender should specify the name of the receiver. We think that these three ways of communication are enough to install natural language processing systems.

3.3. Configuration of DUALS-III

DUALS-III consists of LTB which performs lower-level processing. Based on the LTB tools, DUALS is constructed from modules for high-level processing, such as the problem solver, as shown in Figure 33. Each module of DUALS is controlled and supplied with channels to communicate with other modules, by a process called a LTB Shell. These channels are represented by a thin arrow in Figure 33. Actually we utilized communication paths which were presented as (1) in Figure 32.

DUALS reads a sentence which is inputted at the left top window of , and the following processes are performed on the input sentence, through the channel.

- morphological analysis (LAX);
- syntactic analysis (SAX);
- problem solving (if a question was read);
- sentence generation planning;
- surface sentence generation.

The time and intermediate results for each type of processing are shown in the left center window shown in Figure 34.

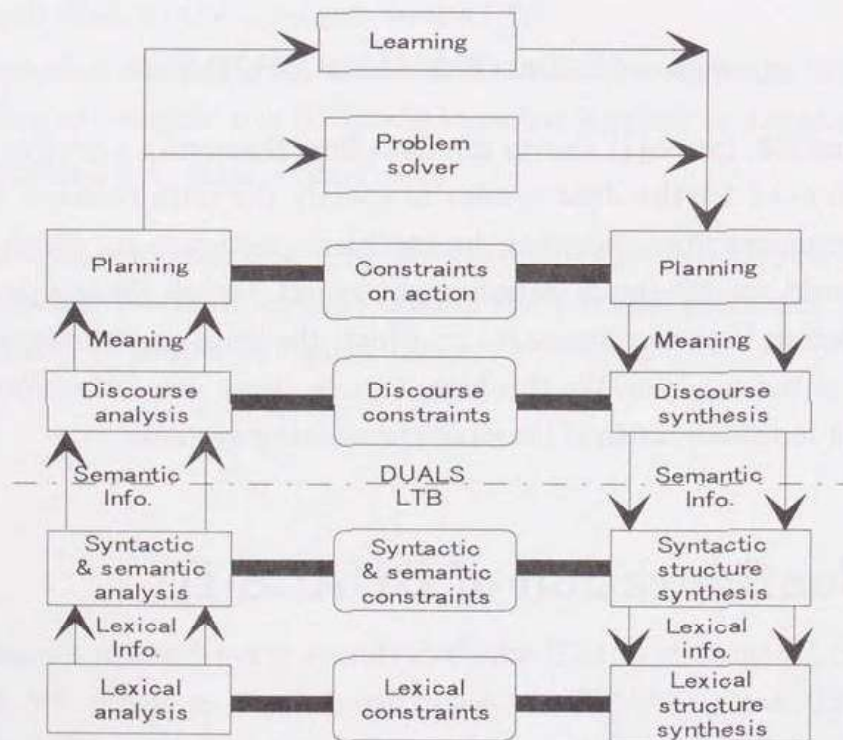


Figure 33 : Configuration of DUALS and LTB

DUALS-III		ICOT 2nd Lab.	
Input Sentence		ElapsedTime (msec.)	
人間が、この地球の上で生き続けていくためには、どうしても自然の恵みに頼らなければならない。私たちが毎日食べる物も、住んでいる家も着る服も、もとはといえば、みな自然界から手にいれた物である。		Solve	
人間がこの地球の上で生き続けていくためにはどうしても自然の恵みに頼らなければならない。		TRS	C-PIn
		SAX	GEN
		LAX	
Output			
TRACE			
上・で 生・き・続・け・て・い・く・た・め・に・は・生・き・続・け・て・い・く・た・め・に・は どうしても/ど・う・し・て・も 自然・の 恵み・に 頼・ら・な・け・れ・ば・な・ら・な・い・\頼・ら・な・け・れ・ば な・ら・な・い			

Figure 34 : Human Interface of DUALS III

Trace of each processes in DUALS is displayed at the bottom window, which is supplied by the LTB tracing facilities.

3.4. Linguistic Knowledge Base

3.4.1. Overview of the Language Knowledge Base

The dictionary [Tanaka] for DUALS-III contains:

- knowledge referred to by some aspects in both analysis and generation(master dictionary and thesaurus);
- semantic knowledge referred by problem solver.

The dictionary forms a tiny experimental linguistic knowledge base together with Japanese grammar made for LTB.

The master dictionary comprises semantic structures representing the meaning and semantic construction rules to connect a word to another word as in the following example.

```
{  
  id_number/verb000080,  
  part_of_speech/verb,  
  inflectional_type/regular,  
  surface/'ATAERU',  
  pronunciation/"ataeru",  
  semantics_index/[verb000080A]  
}.
```

The semantic structures and rules have been developed through some verificational experiments. This knowledge is used in morphological analysis, syntactic analysis, and surface sentence generation, compiled beforehand.

The knowledge used in problem solving may be extracted from input sentences, or may be given in advance. The latter is an independent component of the linguistic knowledge base.

Information contained both in the master dictionary and prepared for problem solving is described in a unified way, to maintain the standardized notation and consistent knowledge of the system.

3.4.2. Master Dictionary

The LTB master dictionary consists of:

- an entry file;
- a word meaning file.

The entry file is a collection of entry records which correspond to word surfaces, and the word meaning file is a collection of word meaning records for each meaning of a word. Generally, one word entry record may correspond to more than one word meaning record.

3.4.3. Future Research for the Linguistic Knowledge Base:

In the development of DUALS-III, our approach was

- first, to establish morphological and syntactic processing as software modules;
- then, on the basis of the software, to implement contextual processing step by step.

However, for the low-level processing to provide high-level processing with enough information, the following reconsideration was required in our language data base.

First, although our master dictionary was expected to be used in both the analysis and the generation parts, and has the minimum information such as lexical categories and voice information for both, there are some discrepancies between the analysis and the generation parts because of the increasing power of discourse processing of DUALS.

We needed to improve the master dictionary so that it would reflect the syntactic and semantic information independently acquired in analysis and generation modules. Up to the end of ICOT, the linguistic database had been focused on morphological information, while syntactic features like voice, aspect, and tense were left for the future studies. Actually the language database was developed after DUALS-III project, and supplied as free software, but there remains many issues around these matters as tense, modality etc.[Tanaka Y.]

Secondly, we also wanted to try to acquire knowledge through learning to enrich the knowledge. Maintaining reservation of consistency of the knowledge base will be the crucial problem. These issues were left for future studies.

3.5. Semantic Description

3.5.1. Overview of CIL in DUALSCIL

In sentence analysis, we need to represent some semantic information together with syntactic information. On this point, CIL plays an important role.

In DUALS, CIL performs:

- Description of Linguistic information;
- Semantic processing;
- System description;

In describing linguistic information, the partially specified term (PST for short), which is used by every module except the problem solver, is known to be very useful. In semantic processing, CIL provides many functions to operate PSTs. In system description, CIL provides a part of programming environment for managing of PST language information.

3.5.2. Partially Specified Terms and Their Operation

A PST is a record-like structure able to represent a network structure, which was introduced in Prolog, to make descriptions of analysis and semantic processing declarative and easy. It can be extended infinitely as long as structural consistency is maintained. In CIL, a PST is written as a set of pairs of attribute and value as follows.

{ name/DUALS, version/3, address/ICOT }

CIL provides a variety of predicates for the attribute and value. In the semantic analysis process, the meaning of each word is combined with the others, forming a larger semantic structure. For this purpose, unification is a basic function and is provided by CIL as follows.

[Execution of Unification]

INPUT> X = {name/DUALS, version/3},
Y = {demo/{conference/FGCS88, place/Tokyo}},
X = Y.

(result of unification)

OUTPUT> X = {name/DUALS, version/3,
 demo/{conference/FGCS88, place/Tokyo}},
Y = {name/DUALS, version/3, demo/{conference/FGCS88, place/Tokyo}}
yes

The predicate that obtains the contents of PST incrementally is given as follows.

[Execution of PST predicate]

INPUT> getRole({name/DUALS, version/3}, A, B).

OUTPUT> A = name, B = DUALS;

A = version, B = 3

yes

Some syntax notations are given for the PST-operating predicate, which are used very often as follows.

[Z is a value of attribute Y in PST X:role(X,Y,Z)]

INPUT> {name/DUALS, version/Z}!version = 3.

OUTPUT> Z = 3

yes

These functions are utilized to construct the meaning of a sentence.

3.5.3. Future Research for the Semantic Description Language

As has been stated, PST provides useful representation and many predicates for operating it. Furthermore, the PTT domain (which is a model of PST) is compact and satisfaction complete relative to partial equality theory. Based on this, the completeness and soundness of the CIL SLD derivation has been proven. In other words, it has been theoretically proven that CIL is an extension of Prolog.

At the end of the middle term of ICOT, we were investigating a type theory and type system, which would enable us to define PST representation and operation in a unified way, for an integrated framework for the three processes performed by CIL. At last, such kind of theoretical work was left for future study. But some type systems were provided as tools in CIL. We intended to extend CIL aiming at a constraint language, and actually CIL was expanded at the end of ICOT to create Quixote [Yasukawa] and the other languages originated at ICOT.

3.6. Morphological and Semantic Analysis

3.6.1. Overview of the Morphological Analyzing System (LAX)

The morphological analyzing program translates a Japanese sentence, which consists of kanji (Chinese characters) and kana (Japanese phonetic characters) which are not separated by blanks, into a sequence of words while looking up a morphological dictionary quickly. This dictionary is converted from the master dictionary in advance.

The morphological analyzer, which can be regarded as having its base on the layered stream technique, which is suitable for both sequential and parallel processing, is realized by the LAX system. The system provides the debugging environment for a morphological analyzer and the converter of the dictionary, which is recently regarded as more and more important for natural language processing. These facilities are actually developed in the

course of LTB activities and at the final stage of ICOT, provided as shareware.

3.6.2. Description Format of the Morphological and Semantic Knowledge

As shown in section 1.3.1., each morpheme has a surface string and information called right- and left- hand feature. The left-hand feature indicates the category which classifies morphemes. The right-hand features of a morpheme stand for the set of categories that can follow it.

Two neighboring morphemes can be concatenated if and only if the left-hand feature of the following morphemes is exactly in the right-hand feature of the preceding morpheme. The details are already given in section 1.3.1.

3.6.3. Special Features of Morphological and Semantic Knowledge

The morphological grammar as morphological and semantic knowledge, which can be regarded as a regular grammar, is based on Morioka's grammar[Morioka]. Figure 35. shows a sample result of morphological and semantic analysis. In this example, the configuration and the meaning for several derivational words such as adjective 「高い」 ("taka-I", high), noun 「高さ」 ("taka-sa", height), transitive verb 「高める」 ("taka-me-ru", raise), intransitive verb 「高まる」 ("taka-ma-ru", rise or increase) are constructed from the word stem 「高」 ("taka"), which has a common surface string and meaning to these words.

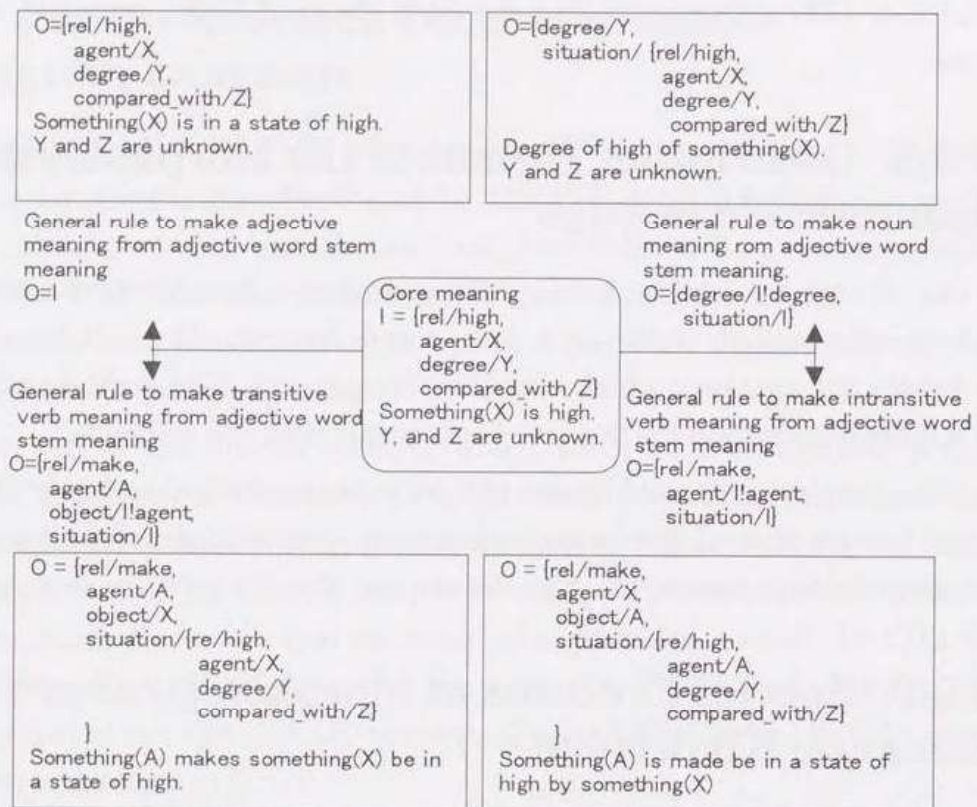


Figure 35 : An Example result of morphological and semantic analysis of DUALS-III

In this method, where the meaning of the word stem 「高」 ("taka") is written, the semantics for each derivational word can be constructed by applying the derivational rules shown in Figure 35. This method therefore provides efficient development of the semantic construction rules. It also makes the outline of the rules used by a problem solver clear because there is no need to write rules for a problem solver that indicates "If something rises then it is in a high state".

As the central meanings of the derivational words are written in only one morpheme, this method reduces the redundancy that is witnessed in traditional methods, in which each derivational word has its own meaning.

Using this method, the morphological analyzer analyzes an input sentence and output several interpretations that are to be reduced to the best one by applying heuristics called the "maximum phrase method" to those interpretations. When ambiguities exist in spite of the heuristics, the

structure corresponding to ambiguities such as those shown in Figure 35 is sent to the syntactic and semantic analyzer SAX without data conversion.

Traditionally, a conversion mechanism is needed to propagate the ambiguous result of the morphological analysis to the syntactic analyzer. However, in DUALS-III, the morphological analyzer is connected with the syntactic analyzer naturally because both of them are based on the same mechanism, called the layered stream mechanism.

3.6.4. Future Research on Morphological Analysis

The morphological analysis program was practical enough even though the result of analysis was ambiguous. It could analyze most sentences that consist of about 30 characters in less than 0.1 seconds. However, the disambiguation mechanism was not very powerful or flexible, so a powerful and flexible mechanism needed to be developed. The grammar rules that morphological knowledge was written with should also be extended from the type 3 grammar to the type2 grammar so as to make the grammar rules flexible.

Classification of morphemes in terms of grammar rules or the description of morphological dictionary was so small that we planned to increase the entries of the dictionary up to 60,000 morphemes and refine the grammar. At last, the grammar rules were published as open software of ICOT².

Although the semantic construction mechanism used in DUALS-III had been effective to a certain extent, it was not yet powerful enough for processing morphemes that have many meanings. We needed to research how to refine this mechanism so that it allowed us to construct semantics flexibly.

² These softwares can be got from the Internet home page located at "<http://www.icot.or.jp/ICOT/IFS/IFS-abst/ifs-catalogue.html>"

3.7. Syntactic and Semantic Analysis

3.7.1. Syntactic Analysis System (SAX)

The syntactic analysis program receives the output of morphological analysis (which is a sequence of words) as input, then determines the dependency-relation among the words, constructing a semantic structure.

This program is developed using the SAX system, and works based on the layered stream algorithm like LAX.

3.7.2. Description of Syntactic and Semantic Knowledge

The grammar of SAX is described in extended DCG. DCG is suitable for representing the tree structures as shown in Figure 36. A DCG rule has a head written on the left-hand side of the rule, and a body, written on the right hand side. The head has one grammatical symbol, while the body is a sequence of grammatical symbols which constitute the head.

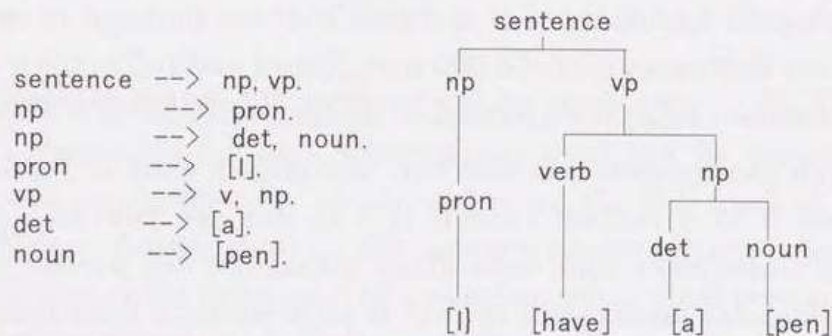


Figure 36 : An Example of DCG rules and its Parsed Tree

3.7.3. Special Features of Syntactic and Semantic Analysis

In the development of DUALS-III, we employed dependency grammar as I mentioned before. In DUALS-III, experimental grammar rules are as follows.

word(Args1),modify_mark,word(Args2) -->

```
word(Args1),(modify_mark;[ ]),word_modify_sequence;),  
word(Args2),{modify(Args1,Args2)}.
```

This rule is a crucial part of the dependency relation and basically all the relations could be written in this form. After utilizing the rules above, we finally got a set of rules which was already introduced in chapter 2.

3.7.4. Handling Ambiguity

The syntactic and semantic analysis parts determine the dependencies of the input word sequence given by the morphological analysis part, using the predicate "modify". When more than one parse is produced, using preference calculation function, the best one is selected [Fukumoto]. This calculation mainly utilizes the distance between a modifying word and modified word. If there is no difference among parses in preference, focusing on the relation of ambiguous meanings, referring to discourse information, and using the constraint logic programming method, the solution is reduced. For example, if it is known from discourse information that 花子("Hanako", a girl's name) was promised something by somebody, the upper two solutions in Figure 24 will be selected. The problem solver, storing these constraints incrementally, can resolve ambiguity. In the existing method, ambiguities have been resolved in parsing with heuristics. With constraint propagation, the analysis part does not need to limit interpretation unnaturally, as is usual[Hasida 88][Hasida 89].

3.7.5. Future Research in Syntactic and Semantic Analysis

The SAX system was extended to allow description of type 0 grammar. However, the development facilities for type 0 grammar were not satisfactory. Because type 0 grammar easily makes a critical pair which causes eternal rewriting, we should investigate a normal form which partially corresponds to type 0 grammar rules. These issues on developing tools for a set of grammar rules in type 0, were partially solved after FGCS 88, through development of new tools for type 0 grammar rules [Fukumoto],

and experimental grammar rules for Japanese sentence analysis[Fukumoto 89]³.

3.8. Problem Solving

3.8.1. Summary of Problem Solving

The problem solver aims at total processing, including anaphora processing, which identifies the object indicated by a pronoun, and problem solving, using the dependency propagation programming technique, devised in research on JPSG [Hasida 87].

A constraint is information about the structure which can be held by some object. We think of discourse processing as constraint processing of a discourse structure. Many constraints related to semantics and pragmatics about discourse are described in a unified notation. No special programs for resolving anaphora or solving problems are necessary.

I think that problem solving is an incubator for new technology of sentence analysis. I and my colleague had many discussion on this theme and tried to visualize the future of sentence analysis. Therefore, I think it is better to touch on this part.

3.8.2. Description of Knowledge for Problem Solver

This section explains the problem solver, using the honorific relation as an example. Assume that the following sentence was uttered in a certain situation[Sugimura 86a]

Situation1:Speaker = Taro, Listener = Hanako

Utterance : "Jiro sama ga Ko rare ta to Tomoko ga Itte i mashi ta.
Tomoko said Jiro came/had come.

Japanese has the constraint that the relation between speaker and listener should be reflected by the form of the predicate of a sentence.

³ At final stage of ICOT, the dependency grammar is served as a free software. The grammar can be got from the Internet located at "<http://www.icot.or.jp/ICOT/IFS/IFS-abst/035.html>".

So, from the above sentence, we know that 'Hanako' is 'Tarou's superior.

This information is acquired through the following processes. Morphological analysis shows that the morpheme 'mashita' has the honorific relation. Next, receiving this result, the syntactic and semantic analysis module judges that the phrase 'imashita' is the predicate of the sentence.

Furthermore, Japanese has another constraint that the relation between speaker and the person mentioned in a conversation is reflected in the expressions which indicate the person. Therefore, the expression 'Tomoko' (a girl's name) uttered by 'Tarou', which has no respect marker, shows that 'Tomoko' is a social equal or inferior of 'Tarou'. This information can be acquired through the previous processing. Whether the phrase 'Jirou-sama ga korareta' is direct speech or indirect speech affects the interpretation of the phrase with respect to the honorific relation. The syntactic analysis part judges that the phrase can be read in two different ways, sending the two interpretations to the problem solver.

Assuming that it is direct speech, we know that 'Jirou' is 'Tomoko's superior, but do not know the relation between 'Tarou' and 'Jirou'. On the other hand, assuming indirect speech, we know that 'Jirou' is 'Tarou's superior, but do not know the relation between 'Tomoko' and 'Jirou'. Therefore, the processing of the next sentence starts, leaving this ambiguity unsolved.

Situation1: Speaker = Taro, Listener = Hanako,
Utterance "Jiro no yatsu Kita no desu ne".
Jiro has come, hasn't he?

From this sentence and the previous one, we know both that 'Jiro' is 'Taro's inferior and that 'Jiro' is 'Tomoko's superior.

As shown in the example, the discourse situations with partial information are fundamental elements in discourse processing and we believe such elements can be applied to many other cases.

Dependency propagation is, like this example, a mechanism which determines a certain structure (in this case the honorific relation), using a constraint related to the structure.

One of the most important areas of research in constraint processing is what structure should be used for representing constraints. Recently, many

interesting results have been obtained, concerning Boolean algebra, for example.

In many cases, one sentence has more than one possible interpretation, some of which may be consistent with an interpretation of another sentence, while some of them may not be.

To obtain the true meaning of a sentence, we need to handle these interpretations correctly, as their relations are complex. Constraint propagation provides a good technique, and was essential for DUALS-III, which aimed at discourse understanding. For example, if a system happens to know some agent of the verb should be person A or B in a discourse, and in another timing a system knows the agent should be B or C, constraint propagation easily concludes that the agent should be B.

There may be two or more constraints which should be handled at one time. If we assign weight to each of them, and control the order in which the constraints are applied, it is easy to handle them.

The DUALS problem solver provided such a function.

3.8.3. Special Features of Knowledge for Problem Solver

The DUALS-III problem solver describes discourse structures assuming that the meaning of a sentence is a constraint[Sugimura 88c]. In other words, it determines discourse structure, storing the meaning of each sentence as a constraint on the discourse structure[Hasida 88].

This discourse structure involves the utterance itself and a model of participants of the conversation. The problem solver treats utterances as a human act, and treats a speech act as a constraint on the discourse structure in the same way as syntactic or semantic constraints. An affirmative sentence produces the constraint that its meaning holds in the real situation (indicated as Situation1), which is a collection of events. A question produces the constraint that the listener should answer the question. In other words, when the system is given a question, the problem solver handles the constraint in which it must answer the question, and therefore answers the question.

Let us provide an example. For the sentence "Human beings need nature to continue to live on the earth.", the structure shown in Figure 37 is produced.

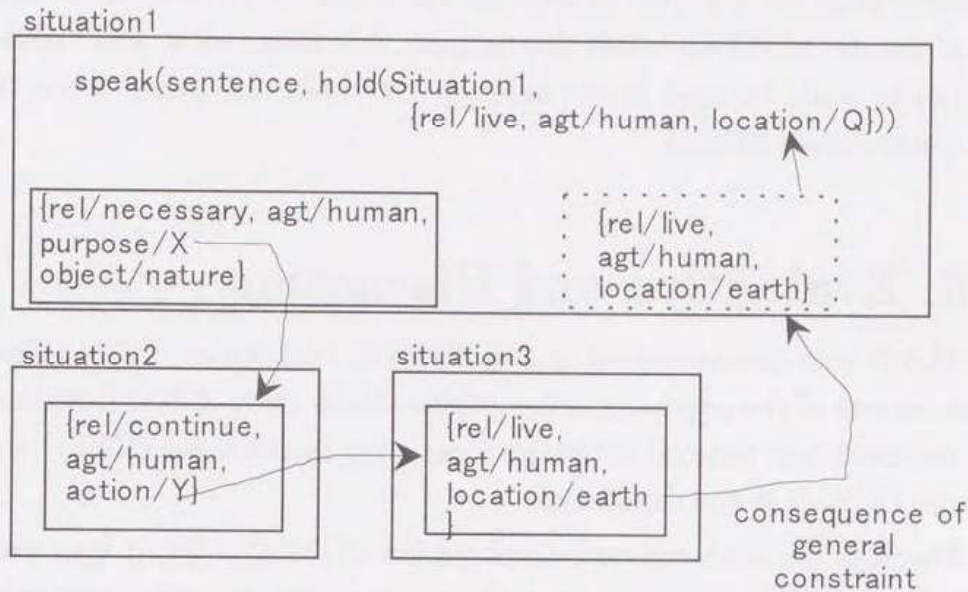


Figure 37 : An Example of Semantic Representation of DUALS-III

Next, when the question "Where do human beings live ?" is given to the system, it searches Situation1, for the fact that human beings live. Using the general constraint that in a situation where something X continues to do something Y presupposes a situation in which X is doing Y, the system produces the fact that human beings live on the earth in Situation1, sending the structure (put in the upper part of the top box) to the sentence generation module.

As shown in this example, constraint propagation allows unified description even for such a complicated discourse processing as constraint processing. In addition, situation theory is a powerful framework which theoretically includes these constraints. Therefore, using constraint propagation and situation theory, the prospect of discourse processing is much better than before, both theoretically and technically.

3.8.4. Future Research on Problem Solving

As already stated, a constraint propagation program makes it possible to determine the structure of a discourse, storing the constraints in it incrementally. At the time of DUALS-III, these constraints were limited to Horn clauses as Prolog. After this project, this framework was expanded so that rules could be read in not only top down but also other ways, to make the system more flexible.

3.9. Evaluation and Discussion

DUALS-II was demonstrated at the FGCS'88 conference, held in December 1988, as one of the products of the intermediate state of ICOT. At that time, the research into natural language processing in ICOT was likely to revolve around DUALS in the final stage.

Through the research and development of DUALS-III, it was confirmed that constraint programming and situation theory efficiently elucidate complicated discourse understanding and acquire the required knowledge.

From the point of analysis part of DUALS, we recognized that sentence analysis fundamentally should have strong relation to the discourse, and it is better for us to make spiral progress stepping on sentence analysis and discourse analysis.

Although DUALS-IV was not made by researchers of ICOT at the last period of ICOT unfortunately, these issues have been mainly discussed by each researcher of DUALS projects[Hasida 89]. Ideas emerged from DUALS project were crystallized as a form of some logic-based languages like cu-Prolog[Tsuda].

One of the most important issues in problem solving is how to build up declarative languages with which we can, not only easily write some class of constraints, but also execute the programs in those languages faster. Experimental software was built and awaiting evaluation by many researchers⁴. From the many evaluations by various researchers, I think we will get new ideas toward language analysis. One of the most important

⁴ There provided many tools like Cu-Prolog, MGPT, Quixote, and CIL as free software of ICOT.

issues is how to merge morphological analysis, syntactic analysis, semantic analysis, pragmatic analysis, and discourse analysis into one framework. There are some trials toward such kinds of tools, but I hope for more competition between knowledge description tools for natural language analysis.

Japanese-English Machine Translation

1.1. Introduction

Japanese-English machine translation is a very difficult task. The main reason for this is that the two languages have very different grammatical structures. In Japanese, the subject, object, and verb are not always in the same order as in English. This makes it difficult for a machine to understand the meaning of a sentence. Another problem is that Japanese has a very rich vocabulary, with many words having multiple meanings. This makes it difficult for a machine to choose the correct meaning for a word in a given context.

Despite these difficulties, there have been many attempts to develop Japanese-English machine translation systems. Some of these systems use statistical methods, while others use rule-based approaches. However, none of these systems has been able to achieve high levels of accuracy. This is because the task of machine translation is very complex, and it requires a deep understanding of the grammar and semantics of both languages. In this paper, we will discuss some of the challenges of Japanese-English machine translation and propose some new approaches to solving these problems.

The first part of the paper is devoted to a review of the literature on the topic. It is found that there is a general consensus that the use of the word 'the' is a common feature of English. However, there is disagreement as to whether this is a grammatical rule or a stylistic choice. The second part of the paper discusses the use of the word 'the' in different contexts. It is found that the use of 'the' is more common in formal writing than in informal writing. The third part of the paper discusses the use of the word 'the' in different parts of the sentence. It is found that the use of 'the' is more common in the subject than in the object.

2. Availability and Discussion

The first part of the paper is devoted to a review of the literature on the topic. It is found that there is a general consensus that the use of the word 'the' is a common feature of English. However, there is disagreement as to whether this is a grammatical rule or a stylistic choice. The second part of the paper discusses the use of the word 'the' in different contexts. It is found that the use of 'the' is more common in formal writing than in informal writing. The third part of the paper discusses the use of the word 'the' in different parts of the sentence. It is found that the use of 'the' is more common in the subject than in the object.

The first part of the paper is devoted to a review of the literature on the topic. It is found that there is a general consensus that the use of the word 'the' is a common feature of English. However, there is disagreement as to whether this is a grammatical rule or a stylistic choice. The second part of the paper discusses the use of the word 'the' in different contexts. It is found that the use of 'the' is more common in formal writing than in informal writing. The third part of the paper discusses the use of the word 'the' in different parts of the sentence. It is found that the use of 'the' is more common in the subject than in the object.

The first part of the paper is devoted to a review of the literature on the topic. It is found that there is a general consensus that the use of the word 'the' is a common feature of English. However, there is disagreement as to whether this is a grammatical rule or a stylistic choice. The second part of the paper discusses the use of the word 'the' in different contexts. It is found that the use of 'the' is more common in formal writing than in informal writing. The third part of the paper discusses the use of the word 'the' in different parts of the sentence. It is found that the use of 'the' is more common in the subject than in the object.

The first part of the paper is devoted to a review of the literature on the topic. It is found that there is a general consensus that the use of the word 'the' is a common feature of English. However, there is disagreement as to whether this is a grammatical rule or a stylistic choice. The second part of the paper discusses the use of the word 'the' in different contexts. It is found that the use of 'the' is more common in formal writing than in informal writing. The third part of the paper discusses the use of the word 'the' in different parts of the sentence. It is found that the use of 'the' is more common in the subject than in the object.

The first part of the paper is devoted to a review of the literature on the topic. It is found that there is a general consensus that the use of the word 'the' is a common feature of English. However, there is disagreement as to whether this is a grammatical rule or a stylistic choice. The second part of the paper discusses the use of the word 'the' in different contexts. It is found that the use of 'the' is more common in formal writing than in informal writing. The third part of the paper discusses the use of the word 'the' in different parts of the sentence. It is found that the use of 'the' is more common in the subject than in the object.

4. Japanese Sentence Analysis in Japanese English Machine Translation

4.1. Introduction

Recently, various kinds of machine translation systems for the commercial market have emerged. Although there still remains the ambitious hope in the market that machines will provide a complete answer, though there is no such complete system, translation systems have begun to be recognized as a useful helper for human translators and general users. At least, a translation system knows so many words which no human can remember that the system can be a useful partner for any user. Therefore, a user friendly interface is crucial to the design of machine translation systems.

In the field of machine translation systems, there are many variants for the realization of the systems [Nagao 86]. One of the most idealistic methods of realization and implementation may be a pivot method in which every language is translated into an ideal intermediate language and from this language, every language can be generated. On the other end, a bi-lingual canned text editing system[Sugimura 84] based on hyper-text and its user interface can be a good representative of machine translation especially in the field of conventional documents like formal letters, legal documents, formal speeches and so on. I do not intend to deny the future of this method,

but it seems early to directly tackle it. Our approach is more near-sighted and focuses on the art of user interfaces[Sugimura 93].

Our machine translation system has three features as follows.

- It facilitates several kind of user interfaces which enable the user to interrupt the process of translation and add their knowledge into the process of disambiguation.
- A transducer is included to carry out the translation process[Kawagoe 90]. Dictionary look up is also fast as the result of the development of unique data retrieval system, and utilization of LAX system[Nakajima].
- A Modification Deciding Network is used to analyze the Japanese modifying relations, which enables us to merge several kinds of syntactic and semantic constraints into one framework[Ishikawa 92].

In this section, user interfaces which allow us to enter our knowledge into the system for the process of disambiguation will be represented in detail[Aoyama 90].

From the research stand point, linguistic phenomena which need the human user to help decide their interpretation are best being deeply studied with the aim of making a new algorithm which ultimately handles the phenomena without help from the user.

Therefore, a machine translation system with conversational mode can be a fruitful system from the commercial view point and for research.

4.2. User Interfaces to get Human Knowledge

Our machine translation system facilitates two dialogue modes to interchange knowledge with the user. The first one is a simple mode to help the system solve a problem for which it is difficult to decide the solution on its own. The second is a detailed mode in which the user can disambiguate all the interpretations of each process of machine translation.

In this section, the detailed mode which includes simple mode interaction will be shown and discussed with the aim of determining the future direction of natural language processing.

4.2.1. Morphological Analysis

In morphological analysis, two kinds of dialogue are implemented. One is the dialogue about contents of the dictionary, and the other is to correct the interpretation of the morphological analysis as follows.

1. Correct the interpretations of the morphological analysis.
2. Registration of new words to the dictionary.
3. Correction of the features of the dictionary.

The first case is effective when the system encounters a problem like in the following example.

Input Sentence: その時雨が屋根を濡らした.

The 1st Result: その・時・雨・が・屋根・を・濡・ら・し・た

At that time, the rain makes the roof wet.

The 2nd Result: その・時雨・が・屋根・を・濡・ら・し・た

That rain in May makes the roof wet.

In such cases, the system asks user to correct the segmentation point using the dialogue interface shown in Figure 38,

The second case resembles a case in a kana-kanji conversion system. Users can register many kind of words using the interface facilitated by the system. These three kinds of registration can be carried out by the user interface as follows.

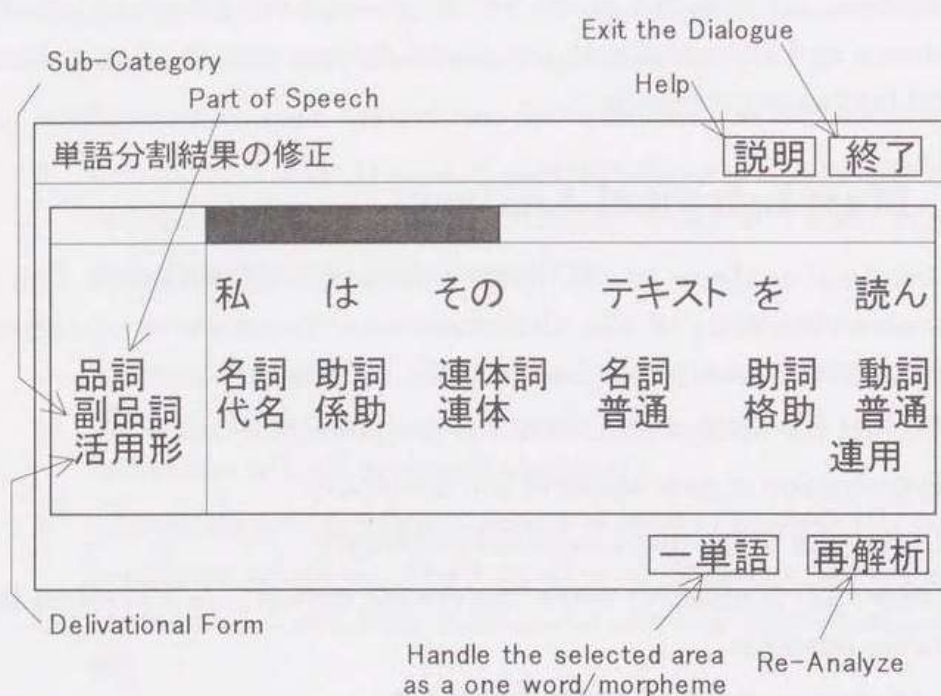


Figure 38 : Human Interface for Morphological Analysis of the Machine Translation System

In Figure 38, there are two buttons for re-analysis and enforced segmentation of the input characters into morphemes. In order to utilize the interface for registration of a new word, the first thing the user should do is to hi-light the area which corresponds to the word, then press the button to register the word. In this case, the user can also register the part of speech, sub-category, and derivational form of the word.

Actually, while it is difficult for the user to register a correct morpheme into the system, this interface plays an important role in correcting sequences of characters which the morphological analyzer cannot handle properly.

Utilizing the dialogue tool, we realized that there will remain the three issues above even though we can make a sophisticated morphological analyzer of quite high quality. We also have to design a morphological analyzer which has strong relation to the sentence analyzer.

4.2.2. Japanese Sentence Analysis

In sentence analysis, there is much information which the system needs the user to correct as follows.

- 1) Disambiguation of multiple meanings of a word.

Ex) 午後は開いている。(Gogo ha Aite iru)

- 1) It opens in the afternoon.
- 2) I am free in the afternoon.

Like the example above, there are many cases in which it is hard for the program to disambiguate the meanings of a word. Another example and a dialogue window is shown in Figure 39

- 2) Disambiguation of dependency structures.

This case was discussed in chapter 2.

- 3) Disambiguation of case markers to be fulfilled.

Ex) 芥川は読んでいた。(Akutagawa wa Yonde ita)

- 1) I have read a book written by Akutagawa.
- 2) Mr.Akutagawa read (it).

In the example above, a human's name Akutagawa can be interpreted as an agent, or an object. In the Figure 40, another example to disambiguate the case and the interface window are shown.

- 4) Disambiguation of the meaning of post positional markers or derivational morphemes.

Ex) 先生は多くの問題を考えられた。(Sensei ha Ookuno Mondai wo Kangae rareta.)

- 1) Our teacher could make many questions.
- 2) Our teacher made many questions. (in polite form)

In this case, the derivational morpheme "rare" can have two

interpretations. One is to display the ability of the agent. The other is to add honorific attitude of the speaker to the teacher.

- 5) Disambiguation of the scope of parallel phrases which usually have parallel case markers or parallel meanings.

EX) 日本の庭と英国の庭園。(Nihon no Niwa to Eikoku no Teien)

Japanese garden and British garden.

日本の庭と寺院の写真。(Nihon no Niwa to Jiin no Shashin)

A photo of a Japanese garden and a temple.

In the example above, although two sentences have the same surface structure which has a sequence of a noun with post positional marker “の(no)”, a noun with post positional marker “と(to)”, a noun with post positional marker “の(no)” and a noun, the interpretations differ from each other because the scope of parallel phrases differs in each sentence.

We can manipulate this information by entering a command into the system through the interfaces as shown in Figure 39. In the figure, the user can fix the multiple meanings of the word.

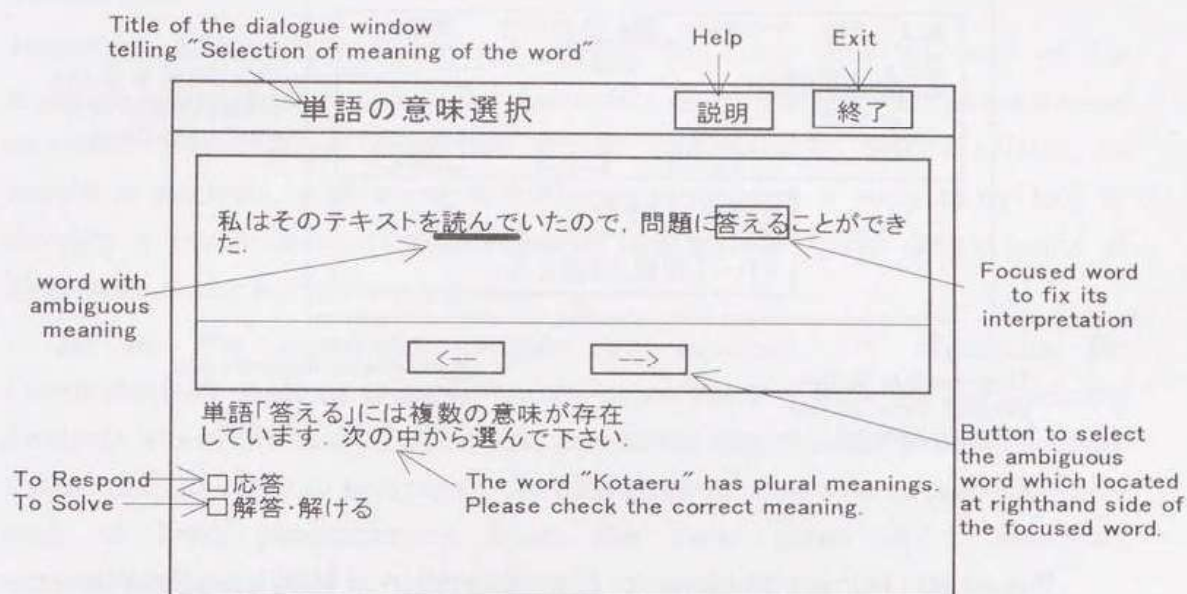


Figure 39 : Human Interface for Word Meaning Disambiguation

Ambiguities in dependency structures, case makers, roles of the case makers, and scope of conjunctions are mutually related. Therefore we made an interface shown in Figure 40 to determine them. Through the interface of the figure, we can not only collect a variety of ambiguities, but also accumulate the example data which will offer a lot of fruitful insights into sentence analysis in future research.

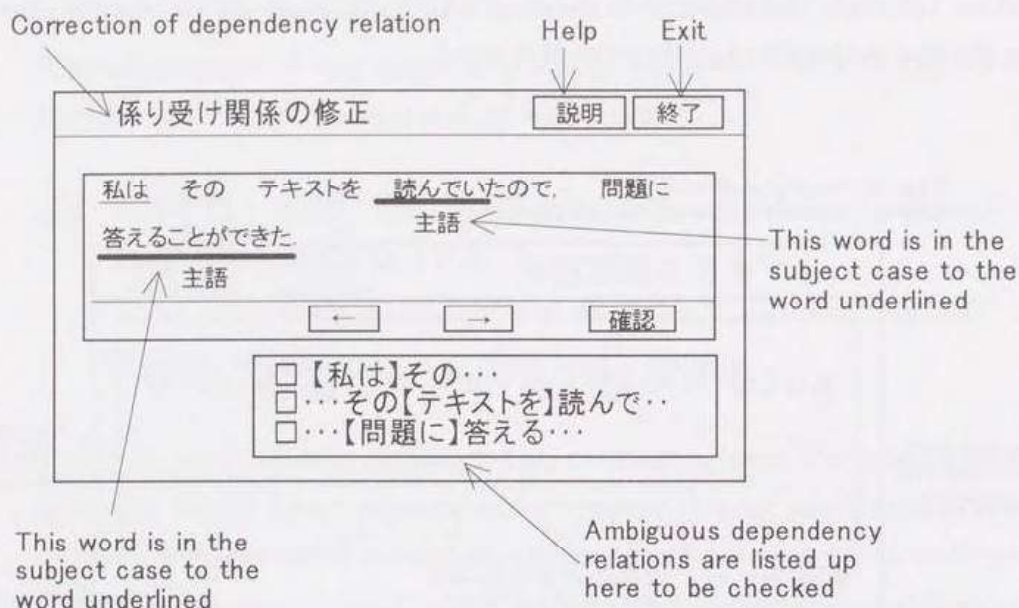


Figure 40 : Human Interface for Disambiguation of Modifying Relations

4.3. Evaluation and Discussion

At first when we start designing the system, we are aiming at a realistic system which will totally realize high reliability in the translation process, in which the system will not eliminate ambiguities with low reliability, and will depend on the human user to decide alternatives which can not be disambiguated without contextual information. After we implemented the facilities to enter the knowledge of the user into the system, the system became more competent at sentence analysis.

However, it is regretful to say that such interactions tend to be regarded as special ones which can be utilized only by specialists who know machine translation systems well. I proposed a system which can be tuned at the user's site, but it was very hard to get agreement from the users to such kind of system.⁵ Actually, the user tends to hate such interaction with the machine

⁵ I had a chance to ask the opinion of the users of machine translation systems at Machine Translation Fair at Osaka in 1993 as a panelist. There was strong opposition.

translation system because the users tend to feel as if they are ordered by the machine which has inferior ability to the user. Such kind of relationship between the user and the machine makes it more difficult to build up a dialogue system which enables the machine to get more knowledge from the human user.

Therefore, I think we should not neglect the facts that the user of the machine translation system will persistently require high quality translation machine. Fortunately, there are many sophisticated works related to sentence analysis, and using a dialogue system as a supporting tool to develop a machine translation system, we should apply many kinds of heuristics to the sentence analysis.

As for the translation engine for analysis, my algorithm for morphological analysis is sufficient to carry out the task. As for sentence analysis, there are many issues left, and as for the transfer process, I think there remains room to rearrange the conventional tree to tree transducer by way of logic programming from the view point of a rewriting system[Sugimura 90a].

5. Conclusion

5.1. Discussion and Future Work

It is a great tribute to the logic programming paradigm that I could continue my research with little deviation from the main issues. From this point of view, logic programming can be regarded as having given me a strong policy to explore in the broad area of natural language processing.

As for morphological analyzers, current technology looks to be saturated because issues left may need a knowledge of sentence analysis or discourse analysis. To realize such kinds of new framework, a new device which enables us to handle linguistic constraints more freely without considering the modularity of the implementation has to be developed [Hasida 87].

As for sentence analysis, I dare say that we could not attain all the issues in sentence analysis so that we had to continue laborious work to collect the actual material. My framework enables us to write linguistic constraints declaratively and simply. Therefore, we can separately discuss the purely linguistic issues and technological issues.

As for the linguistic issues of sentence analysis, I think we have to handle voice, aspect, and tense [Teramura], as issues about which much study still needs to be carried out in regard to machine translation systems. We need to collect data on anaphoric expressions [Kameyama] which are very important in Japanese-English translation systems. There should be some model to handle intention which appears in the inflectional morpheme.

As to the technological issues which translates and executes the linguistic knowledge, more feasible tools will be beneficial for the researchers of this

area. These days, personal computers have almost the same ability as workstations, so that we should develop our tools on PCs with popular languages. On the other hand, the age of parallel processing is emerging from the field of data base systems, and this trend will be a main stream in the decade. These factors will support a parallel language and its application in near future.

To summarize my work concerning logic based Japanese sentence analysis, I will make a list of my work as follows. Beginning from the research and development of morphological analyzer, I hoped I could contribute to the research to handle linguistic constraints all in logic. Having logic programming as research background, I think we should tackle the research areas which are displayed in gray color in the list.⁶

Actually there are many works which tackle many of the hard problems in the area displayed in gray color. I hope good competition between different kinds of approaches may open the door to the next decade of natural language research.

	Morphological Analysis	Sentence Analysis	Discourse Analysis
Framework to write knowledge (Theory)	LAX Algorithm	Extended CFG (CSG)	Constraint Satisfaction
Tool to manipulate the knowledge	PSI, WS, PC	PSI	none
Accumulation of Knowledge	70,000 words 99.84% precision	experiments by DUALS-III	experiments by DUALS-III

Natural language processing can be regarded as a science comparable in size to space technology. In the old days, human beings could not grasp all the celestial systems and made a lot of myth and rough partial theory which could be applied only to some parts of the sky. Natural language may resemble the celestial system in that we cannot grasp the whole phenomena even though we utilize language all the time.

⁶ As am in charge of a project of machine translation systems for market use, I am actually struggling to build a new framework using large bi-lingual text base, but this work is beyond this thesis and should be discussed in another paper.

The main factor which promotes modern scientific research on space was detailed work on the observation of stars. Tycho Brahe, a Danish astronomer gave a tremendous amount of background data to Kepler who founded a crucial theory about the solar system. Sometimes a partial observation of natural phenomena like a dropping apple could lead to a big theory which explains almost all the phenomena of the sky.

Even about ten years ago, it was very hard for us to process actual natural language texts by computer. But now, we are entering the new age of utilizing actual text on the computer. This fact will give us a chance to analyze the actual material, so that at last we can apply the effective methodology of modern science to natural language processing.

6. List of Publications

6.1. List of Major Publications

- [Sugimura 84] "Teikei Bunsho to Sono Bunsho Sakusei Sien System heno Ouyou ("Document Template Compiler and its Application to Document Creating System" in Japanese)", Ryoichi Sugimura, Junichi Kubota, Yoshimi Mukai, and Makoto Tatebayashi, Natural Language Processing Symposium of IPSJ, pp.43-48, 1984
- [Sugimura 86a] "Japanese Honorifics and Situation Semantics", Ryoichi Sugimura, Proceedings of International Conference on Computational Linguistics 86, pp. 507-510, 1986
- [Sugimura 86b] "Nihongo no Taigu Hyougen to Joukyou Imiron (Japanese Honorifics and Situation Semantics in Japanese)", Ryoichi Sugimura, Computer Software, Vol.3, No.4, pp.46-54, October 1986
- [Sugimura 87a] "Constraint Analysis on Japanese Modifying Relations", Ryoichi Sugimura, Hideo Miyoshi, and Kuniaki Mukai, Proceedings of Second International Workshop on Natural Language Understanding and Logic Programming, 1987,
- [Sugimura 88a] "Constraint Analysis on Japanese Modification", Ryoichi Sugimura, Hideo Miyoshi and Kuniaki Mukai, Natural Language Understanding and Logic Programming, II, pp. 93-106, V.Dahl and P.Saint-Dizier (Editors), Elsevier Science Publishers B.V. (North-Holland), 1988

- [Sugimura 88b] "Ronri-gata Keitaiso Kaiseki LAX ("Logic based Morphological Analyzer LAX" in *Japanese*)", Ryoichi Sugimura, R., Akasaka, K., Kubo, Y., Matsumoto, Y., Logic Programming Conference, pp.213-222, April, 1988
- [Sugimura 88c] "Ronri-Gata Bunpo ni okeru Seiyaku Kaiseki (Constraint Analysis on Logic Grammars (in Japanese)", Ryoichi Sugimura, Proceedings of the 2nd Annual Conference of Japanese society for Artificial Intelligence , pp.427-430, 1988, a **Prized Paper**
- [Sugimura 89a] "Logic Based Lexical Analyser LAX", Ryoichi Sugimura, Koji Akasaka, Yukihiro Kubo, Yuji Matsumoto, F. Furukawa, H. Tanaka, T.Fujisaki(Editors), Lecture Notes in Artificial Intelligence, Vol. 383, pp. 188-216, Springer-Verlag, 1989
- [Sugimura 89b] "Natural language processing in the experimental discourse understanding system DUALS III", Ryoichi Sugimura, Kouji Akasaka, Yuichi Tanaka, Koichi Hasida, Kuniaki Mukai, and Yukihiro Kubo, Logic and Logic Grammars for Language Processing, Patrick Saint-Dizier and Stan Szpakowicz (Editors), Ellis Harwood Series in Artificial Intelligence, pp. 185-204
- [Sugimura 89c] "Danwa Rikai Jikken System DUALS Dai-3 han niokeru Sizengengo Shori ("Naturaln Lanuage Processing in the Discourse Understanding System DUALS III" in Japanese) ", Ryoichi Sugimura, Koichi Hasida, Yuichi Tanaka, and Kuniaki Mukai, Artificial Intelligence, Vol4, No.3, pp285-296, 1989
- [Sugimura 89d] "A Software Environment for Research into Discourse Understanding Systems", Ryoichi Sugimura, Koichi Hasida, Koji Akasaka, Toshiyuki Okunishi, Yukihiro Kubo, Koji Hatano, and Yuichi Tanaka, Proceeding of FGCS'88, pp.285-295, 1988
- [Sugimura 90a] "Logic based Tree to Tree Transducer", Ryoichi Sugimura, Mutsumi Kawagoe, German-Japanese Workshop on Logic and Natural Language, Kyoto, October, 1990
- [Sugimura 93] "Nichi-Ei Honyaku Shien System ("Japanese English Machine Aided Translation System" in Japanese), Ryoichi Sugimura, National Technical Report, Vol.39, pp.80-88, 1993

6.2. List of Other Publications

- [Sugimura 86c] "Logic Programming wo Base nishita Sizengengo System no Hikaku ("Comprison of Natural Language Processing System based on Logic Programming" in Japanese)", Ryoichi Sugimura, Yuji Matsumoto, Toshiyuki Okunishi, and Hozumi Tanaka, IPSJ-NL, 57-2, September, 1986
- [Sugimura 86d] "Koubun Kaiseki System SAX no CIL niyoru Jitsugen (Implementation of Parsing System SAX with CIL in Japanese)", Proceedings of the 33rd Conference of Information Processing Sociey of Japan, pp.1799-1800, October 1986
- [Sugimura 87b] "DUALS notameno Chishiki Kijutsu Gengo CIL no Hyouka (Evaluation of Knowledge Description Language CIL for DUALS in Japanese)", Fifth Generation Computer Symposium, Jun 1987
- [Sugimura 87c] "Heiretsu Keitaiso Kaiseki ,(Parallel Morphological Analysis *in Japanese*), Ryoichi Sugimura, Yuji Matsumoto, Japanese Society of Software Science and Technology, Logic and Natural Language, March, 1987
- [Sugimura 87d] "Heiretsu Keitaiso Kaiseki System LAX no Jitsugen" ("Implementation of parallel morphological analyzing system LAX" in Japanese), Ryoichi Sugimura, Koji Akasaka, Yuji Matsumoto, Proceedings of the 35 the Conference of Information Processing Society of Japan, pp.1323-1324, October, 1987
- [Sugimura 87e] "Constraint ni Motoduku Nihongo no Kakariuke Kaiseki (Analysis of Japanese Modifying Relation based on Constraint in Japanese)", Ryoichi Sugimura, Kuniaki Mukai, Hideo Miyoshi, Proceedings of the 35th Conference of Information Processing Society of Japan, pp.1331-1332, 1987
- [Sugimura 88d] "Ronri-gata Bunpo nioketu Seiyaku Kaiseki (Constraint Analysis in Logic Grammar in Japanese)", Ryoichi Sugimura, IPSJ-NL, 67-2, 1988

- [Sugimura 88e] "Hannyo Nihongo Shorikei LTB no Kousei (Structure of General Purpose Japanese Processing System LTB)", Ryoichi Sugimura, Koji Akasak, Toshiyuki Okunishi, Yukihiro Kubo, Koji Hatano, and Yuichi Tanaka, Proceedings of the 37th Conference of Information Processing Society of Japan, pp.1072-1073, 1988
- [Sugimura 89f] "Nihon-go no Kaiseki-wo Heiretsu Shori-suru" (Parallel Analysis of Japanese in Japanese)", Ryoichi Sugimura, Yuji Matsumoto, Suri-Kagaku, No. 313, pp34-41, July, 1989
- [Sugimura 89f] "Logical dependency grammar and its constraint analysis", Ryoichi Sugimura, ICOT TM No.679, 1989
- [Sugimura 89g] "Ronri-gat Bunpo ni yoru Izon Kozo Kaiseki ("Dependency Analysis on logic grammars" Japanese)", Ryoichi Sugimura, Fumiyo Fukumoto, Danwa Rikai Model to Sono Ouyou Sympsium (Symposium on Discourse Understanding Model and Its Applications), pp.89-100, 1989
- [Sugimura 90b] "Imi Code Taikei no Jidou Seisei (Automatic Generation of Semantic Code Trees in Japanese)", Ryoichi Sugimura, Koji Kakigahara, Masahiko Ishikawa, Mutsumi Kawagoe, and Shoichi Aoyama, Japanese Society of Artificial Intelligence, SIG-FAI-9001-7(6/5), pp.65-73, 1990
- [Sugimura 90c] "Imi Code Taikei no Jidou Seisei (Automatic Generation of Semantic Code Trees in Japanese)", Ryoichi Sugimura, Koji Kakigahara, Masahiko Ishikawa, Mutsumi Kawagoe, and Shoichi Aoyama, Japanese Society of Artificial Intelligence, IPSJ NL 78-4, pp.25-32, 1990

6.3. Related work

- [Aho] "The theory of Parsing, Translation, and Compiling", A.V.Aho, and J.D.Ullman, vol. 1 and 2, Prentice-Hall, 1972
- [Aizawa] "Mechanical Translation System of Kana Representations to Kanji-kana Mixed Representations" (in Japanese), Aizawa, T. and Ebara, T., NHK Gijutsu-Kenkyu, Vol.25, No.5, pp.23-60, 1973

- [Akasaka 87] "Chikuji Shorikei-jo deno LAX no Mondaiten to Sono Kaiketsuhou (Issues of LAX system on procedural processing system, and its solution in Japanese)", Koji Akasaka, Ryoichi Sugimura, Yuji Matsumoto, Proceedings of the 35th conference of Information Processing Society of Japan, pp.1355-1356, October, 1987
- [Aoyama 90] "Taiwa Honnyaku no Ichi Houshiki ni tsuite", Shoichi Aoyama, Masahiko Ishikawa, Koji Kakigahara, Tatsuya Katoh, and Ryoichi Sugimura, NLC90-14, pp17-24, 1990
- [Barwise] "Situations and Attitudes", Jon Barwise and Jon Perry, MIT Press, Cambridge, 1983.
- [Chikayama] "ESP Reference Manual", Takashi Chikayama, Technical Report 044, ICOT, 1984
- [Colmerauer] "Prolog-II: Reference Manual and Theoretical Model", A. Colmerauer, Internal Report, Group Intelligence Artificielle, Universitaet Aix-Marseille II, 1982
- [Dincbas] "Extending equation solving and constraint handling in logic programming", M. Dincbas, H.Simonis and P.V. Hentenryck, ECRC Internal Report IR-LP-2203, 1987
- [Fujisaki 89] "Probabilistic Parsing Method for Sentence Disambiguation", T.Fujisaki, F. Cocke, J.Black, and T.Nishino, Proceedings of International Workshop on Parsing Technologies, CMU, pp.85-94, 1989
- [Fukumoto] "A debugging tool based on a dependency structure analysis built around the SAX system", Fumiyo Fukumoto, Kouji Akasak, Naoto Hirota, and Satoshi Itoh, ICOT TM-837
- [Fukumoto 89] "Dependency Structure Analysis in SAX - composition of syntactic and semantic information-", Fumiyo Fukumoto, Ryoichi Sugimura, TM-773
- [Gallier] "Logic for Computer Science - Foundations of Automatic Theorem Proving", Jean H. Gallier, pp. 12-13, Haper & Row, Publishers, New York 1986
- [Gazdar] "Generalized Phrase Structure Grammar", Gerald Gazdar, Cambridge, Harvard University Press, 1985

- [Hasida 87] "Dependency Propagation : Unified theory of sentence comprehension and generation", Koichi Hasida, Proceedings of the 10th IJCAI, pp.664-670, 1987
- [Hasida 88] "DUALS-II niokeru Danwa Shori (Discourse Processing in DUALS III in Japanese)", Koichi Hasida, Ryoichi Sugimura Yuichi Tanaka, Kuniaki Mukai, Proceedings of 37th conference of Information Processing Society of Japan, pp.1121-1122, September 1988
- [Hasida 89] "Danwa Rikai Jikken System DUALS (Discourse Understanding System DUALS III in Japanese)", Koichi Hasida, Ryoichi Sugimura, Yuichi Tanaka, Journal of Information Processing Society of Japan, Vol.30, No. 10, pp.1250-1257, 1989
- [Ishikawa 92] "Natural Language Analysis Using a Network Model - Modification Deciding Network", Masahiko Ishikawa, Ryoichi Sugimura, 4th International Conference on Theoretical and Methodological Issues in Machine Translation, pp.55-66, 1992
- [Kameyama] "Centering in Japanese; Basic Observation", Megumi Kameyama, PhD Thesis, Department of Linguistics, Stanford University, 1984
- [Kamiwaki] "Jisho no TRIE Kozo-ka to Jukugo Shori (Idiom Handling Using the TRIE Structure Dictionary in Japanese)", T. Kamiwaki, and Hozumi Tanaka, Proceedings of the Logic Programming Conference '85, pp.329-340, 1988
- [Kaplan & Bresnan] "Lexical-Functional Grammar: A Formatl System for Grammatical Representation", R.M. Kaplan, J. Bresnan., The Mental Representation of Grammatical Relations (Joan Bresnan,ed.), pp.173-281, MIT Press, Cambridge MA, 1982
- [Kato 92] "Honyaku Katei niokeru Kouzou Henkan no Kousokuka (High Performance in Transfer phase of Machine Translation System in Japanese)", Tatsuya Katoh, Katsumi Tokuda, Mutsumi Kawagoe, Shoichi Aoyama, Kimikazu Fujita, and Ryoichi Sugimura, NLC 92-3, pp17-23, 1992

- [Kawagoe 90] "Ki Kozo Henkan Gengo Allegro -C (Tree to Tree Transducer Allegro-C Japanese)", Mutsumi Kawagoe, Shoichi Aoayama, Ryoichi Sugimura, IPSJ 41-3, pp.152-153, 1990
- [Kimura 86] "Danwa Rikai Jikken System DUALS Dai 2 han no Sekkei to Jissou", Kazuhiro Kimura, Ryoichi Sugimura, Takashi Takizuka, and Kuniaki Mukai, JSSST Taikai, 1986
- [Kubo 88] "LTB-Keitaiso Kaiseki System LAX no Kaihatsu Kankyou (Development Circumstances for Morphological Analyzing System LAX under LTB in Japanese)", Yukihiro Kubo, Takamasa Yoshizumi, Hiroshi Sano, Koji Akasaka, and Ryoichi Sugimura, Proceedings of the 37th conference of Information Processing Society of Japan, pp.1078-1079, 1988
- [Kubo 89] "Construction of Semantic Structure of words in terms of the Morphological and Semantic Analyser LAX", Yukihiro Kubo, ICOT TM-774
- [KUBOTA 94] "Katakana Hyouki no Touitsu Houshiki (A Method of Detecting KATAKANA Variants in a Document in Japanese)", Jun'ichi Kubota, Yukie Shoda, Masahiro Kawai, Hirofumi Tamagawa, and Ryoichi Sugimura, Journal IPSJ, Vol.35, No.12, pp.2745-2751, 1994
- [Matsumoto 86] "Parsing System SAX based on logic programming language", Yuji Matsumoto, Ryoichi Sugimura, Computer Software, Vol.3, No.4, pp.4-11, (also in ICOT-TR No.200), 1986
- [Matsumoto 87] "A Parsing System Based on Logic Programming", Yuji Matsumoto, Ryoichi Sugimura, Proc. of International Joint Conference on Artificial Intelligence (IJCAI) 87, 1987
- [Matsumoto 88] "Koubun Kaiseki System SAX no tamenno Bunpo Kijutu Gengo", Yuji Matsumoto, Ryoichi Sugimura, Journal of Fifth Conference of Japanese Society of Software Science and Technology, pp.77-80, September 1988
- [Mizutani 1983] "Taigu hyougen no Shikumi (Structure of Honorific Relations in Japanese)", Shizuo Mizutani, Unyo 1 (Pragmatics 1), Asakura, pp.158-178, 1983
- [Morioka] "Goi no Kousei (Formation of a Vocabulary in Japanese)", Gendai-go Kenkyuu Series 1, Meiji-Shoin, 1987

- [Mukai 88] "Partially Specified Term in Logic Programming for Linguistic Analysis", Kuniaki Mukai, Proceedings of Fifth Generation Computer Symposium '88, ICOT, 1988
- [Nakajima] "TRIE Kouzou to Graph Stack wo mochiita Nihongo Keitaiso Kaiseki (Japanese Morphological Analyzer using TRIE structure and Graph Stack in Japanese)", Akiko Nakajima, and Ryoichi Sugimura, Proceedings of 39th conference of Information Processing Society of Japan, pp.589-590, October 1989
- [Nagao 83] "Gengo Kougaku (Technology of Language in Japanese)", Makoto Nagao, Shoukou-do, pp.91, 1983
- [Nagao 86] "Kikai Hon'yaku ha Dokomade Kanou-ka", Makoto Nagao, Iwanami, 1986
- [Okunishi 87] "Comparison of Logic Programming Based Natural Language Parsing Systems", Toshiyuki Okunishi, Ryoichi Sugimura, and Yuji Matsumoto, Proc. of 2nd International Workshop on Natural Language and Logic Programming 87, pp. 90-102, 1987
- [Okunishi 87] "Comparison of Logic Programming Based Natural Language Parsing Systems", Toshiyuki Okunishi, Ryoichi Sugimura, and Yuji Matsumoto, Proc. of 2nd International Workshop on Natural Language and Logic Programming II, North-Holland, pp.1-14, 1987
- [Pereira & Warren] "Definite Clause Grammars for Language Analysis – a survey of the Formalism and a Comparison with Augmented Transition Networks", F. Pereira and D.H.D. Warren, Artificial Intelligence, Vol.13, pp.231-278, 1980
- [Pollard] "Generalized Phrase Structure Grammars, Head Grammars, and Natural Language", Carl Pollard, Doctoral Dissertation, Stanford University, 1984
- [Sakai] "Boolean Graebner Bases", Ko Sakai, ICOT Technical Memo No.488, 1988
- [Sano 88] "Go-Kousei ni Motodoku Keitaiso Kaiseki (Morphological Analysis with delivation and inflection in Japanese)", Hiroshi Sano, Koji Akasaka, Yukihiro Kubo, and Ryoichi Sugimura, Proceedings of the 36 Conference of Information Processing Society of Japan.

- [Sells] "Lectures on Contemporary Syntactic Theories - An Introduction to Government-Binding Theory, Generalized Phrase Structure Grammar, and Lexical-Functional Grammar", Peter Sells, CSLI, Stanford University, 1985
- [Susaki] "Multi PSI ni okeru Heiretsu Koubun Kaiseki Program PAX no Jitsugen oyobi Hyouka (Implementation and Evaluation of Parallel Parsing System PAX intalled on Multi-PSI in English)", Kasumi Susaki, Hiroyuki Sato, Ryoichi Sugimura, Koji Akasaka, Kazuo Taki, Naoto Hirota, Proceedings of Japanese Society of Software Science and Technology, 1989
- [Takizuka 87] "Shizen Gengo Bun no Kaiseki to Seisei Touitsu Bunpo ni Kansuru Ichi Houshiki (A method to unite the grammar of analysis and synthesis of natural language in Japanese)", Takashi Takizuka, Koji Akasaka, Hiroshi Sano, Shinichi Shigenaga, Proceedings of the 35th conference of Information Processing Society of Japan, pp.1365-1366, October 1987
- [Takizuka 88a] "LTB Master Jisho no Kousei (Structure of LTB Master Dictioanry in Japanese)", Takashi Takizuka, Ryoichi Sugimura, Yuichi Tanaka, Conference on Logic and Natural Language of Japanese Society of Software Science and Technology Marchi 1988.
- [Takizuka 88b] "LTB Shell no Kousei (Structure of LTB Shell in Japanese), Takashi Takizuka, Ryoichi Sugimura, Proceedings of 37th conference of Information Processing Society of Japan, pp.1074-1075, September 1989
- [Tanaka] "Shizengengo Kaiseki no Kiso", Hozumi Tanaka, Sangyou Tosho, 1989.
- [Tanaka 79] "Preictive Control Parser:Extended LINGOL", Hozumi Tanaka, Taisuke Sato, and F. Motoyoshi, IJCAI '79, pp868-870, 1979
- [Tanaka, Y] "Natural Langaage Processing", Yuichi Tanaka, Proceedings of the International Conference on Fifth Generation Computer Systems 1992, pp.155-165
- [Teramura] "Nihongo no Syntax to Imi I,II,III", Hideo Teramura, Kuroshio Shuppan, Tokyo, 1982

- [Tomita] "Efficient Parsing for Natural Language", Masaru Tomita, Kluwer Academic Publishers, 1986
- [Tsuda] "cu-Prolog for Constraint-Based Grammar", Hiroshi Tsuda, Proceedings of the International Conference on Fifth Generation Computer Systems 1992, pp.347-356, 1992
- [Ueda] "Guarded Horn Clauses", Kazunori Ueda, Technical Report 103, ICOT, 1985
- [Yamazaki 88] "Koubun Kaiseki System SAX no Debug Kankyou", Shigeichiro Yamazaki, Ryoichi Sugimura, Yuji Matsumoto, Journal of 2nd Conference of Japanese Society of Artificial Intelligence, pp.411-414, 1988
- [Yasukawa 92] "Objects, properties, and Modules in Quixote", Hideki Yasukawa, Hiroshi Tsuda, and Kazumasa Yokota, Proceedings of the International Conference on Fifth Generation Computer Systems 1992, pp.257-268
- [Yokoi 86] "Research Activities on Natural Language Processing of the FGCS Project", Toshio Yokoi, Kuniaki Mukai, Hideo Miyoshi, Yuichi Tanaka, and Ryoichi Sugimura, Fall Joint Computer Conference, IEEE, 1986
- [Yoshida 72] "Syntax Analysis of Japanese Sentence Base on Kakariuke Relation between Two Bunsetsu", Sho Yoshida, IECE, Vol.55-D No.4, pp.238-244, 1972
- [Yoshimura 83] "Morphological Analysis of Non-marked-off Japanese Sentence by the Least BUNSETSU's Number Method", Kenji Yoshimura, Toru Hitaka and Sho Yoshida, Journal of Information Processing Society of Japan, Vol.24, No.1, pp.40-46, 1983
- [Yoshimura 89] "Morphological Analysis of Japanese Sentences Containing Unknown Words", Kenji Yoshimura, Mitsuno Takeuchi, Kenzo Tsuda and Kosho Shudo, Journal of Information Processing Society of Japan, Vol.30, No.3, pp.294-301, 1989

7. Appendix A

Appendix A

lex_P2([C | W], Result):-

dict(C, W, 0, [[0, end([word, top_of_sentence]), nil]], out),
mawaru(W, 1, Out, Result),

mawaru([], NO, IN, IN).

mawaru([W | R], NO, IN, out):-

dict(W, R, NO, IN, O1), NO1 is NO + 1,
mawaru(R, NO1, O1, Out).

% dict Part

dict(い, C_Words, ID_NO, Instr, Ostr0) :-

IDNew is ID_NO + 1,
filstr(Instr, ID_NO, Ostr0, Ostr1, StObj),
filcon(StObj,
[胃前方, IDNew, 胃(体言語基), 胃後方], Ostr1, Ostr2),
(C_Words = [C | Words],
い(C, Words, ID_NO, StObj, Ostr2, Ostr3),
Ostr2 = Ostr3),
sweep(Ostr0, Ostr3, [[IDNew, 1, 未登録語(い) | StObj]]).

dict(め, C_Words, ID_NO, Instr, Ostr0) :-

IDNew is ID_NO + 1,
filstr(Instr, ID_NO, Ostr0, Ostr1, StObj),
filcon(StObj,
[め前方, IDNew, め(動詞派生接辞), め後方], Ostr1, Ostr2),
(C_Words = [C | Words],


```

め(C, Words, ID_NO, StObj, Ostr2, Ostr3),
Ostr2 = Ostr3),
sweep(Ostr0, Ostr3, [[IDNew, 1, 未登録語(め)|StObj]]).

dict(る, C_Words, ID_NO, Instr, Ostr0) :-
IDNew is ID_NO + 1,
filstr(Instr, ID_NO, Ostr0, Ostr1, StObj),
filcon(StObj,
[る前方, IDNew, る(ウ系弱変化活用助辞), る後方], Ostr1, Ostr2),
(C_Words = [C|Words],
る(C, Words, ID_NO, StObj, Ostr2, Ostr3),
Ostr2 = Ostr3),
sweep(Ostr0, Ostr3, [[IDNew, 1, 未登録語(る)|StObj]]).

dict(な, C_Words, ID_NO, Instr, Ostr0) :-
IDNew is ID_NO + 1,
filstr(Instr, ID_NO, Ostr0, Ostr1, StObj),
filcon(StObj,
[な前方, IDNew, な(終助詞), な後方], Ostr1, Ostr2),
(C_Words = [C|Words],
な(C, Words, ID_NO, StObj, Ostr2, Ostr3),
Ostr2 = Ostr3),
sweep(Ostr0, Ostr3, [[IDNew, 1, 未登録語(な)|StObj]]).

% Process for character which is not in the dictionary
dict(C,RR, ID_NO, Instr, Ostr0) :-
IDNew is ID_NO + 1,
filstr(Instr, ID_NO, Ostr0, Ostr1, StObj),
sweep(Ostr0, Ostr1, [[IDNew, 1, 未登録文字(C)|StObj]]).

%Sub Processes
い(た, C_Words, ID_NO, StObj, Ostr0, Ostr3) :-
IDNew is ID_NO + 2,
filcon(StObj,
[痛前方, IDNew, 痛(形容語基), 痛後方], Ostr0, Ostr1),
filcon(StObj,
[傷前方, IDNew, 傷(用言語基), 傷後方], Ostr1, Ostr2),
(C_Words == [],

```

```

Ostr2 = Ostr3,
C_Words = [C | Words],
いた(C, Words, ID_NO, StObj, Ostr2, Ostr3)).

い(,_,_,_,X,X).

いた(め, C_Words, ID_NO, StObj, Ostr0, Ostr2) :-
    IDNew is ID_NO + 3,
    filcon(StObj,
        [炒め前方, IDNew, 炒め, 炒め後方], Ostr0, Ostr1),
    filcon(StObj,
        [板目前方, IDNew, 板目, 板目後方], Ostr1, Ostr2).

いた(,_,_,_,X,X).
め(,_,_,_,X,X).
る(,_,_,_,X,X).
な(,_,_,_,X,X).

filstr([ ], _, X, X, [ ]) :- !.
filstr([[ID | R] | RR], ID, Str0, Str1, [R | R1]) :- !,
    filstr(RR, ID, Str0, Str1, R1).
filstr([[ID | R] | RR], IDN, [[ID | R] | Str0], Str1, R1) :-
    ID > IDN, !,
    filstr(RR, IDN, Str0, Str1, R1).
filstr([[ID | R] | RR], IDN, Str0, Str1, R1) :-
    ID < IDN, !,
    filstr(RR, IDN, Str0, Str1, R1).

filcon([ ], _, X, X) :- !.
filcon([[KS | W] | Data], [CMS, Id, HC, US],
    [[Id, US, HC, W | RO] | Strtail], Strtail) :-
    check(KS, CMS), !,
    filcon1(Data, CMS, RO).
filcon([_ | Data], CMSHC, RO, Stt) :- !,
    filcon(Data, CMSHC, RO, Stt).

```



```
filcon1([ ],_, [ ]) :- !.
```

```
filcon1([KS|W]|Data],CMS,[W|R]) :-
```

```
    check(KS, CMS),!,
```

```
    filcon1(Data, CMS,R).
```

```
filcon1([_|Data], X, Y) :- !,
```

```
    filcon1(Data, X, Y).
```

```
sweep(X,Y,Z) :- X == Y, X = 1.
```

```
sweep(X,Y,Z) :- nonvar(X),!.
```

```
check(end([文節, 文頭]), 胃前方).
```

```
check(end([文節, 文頭]), 痛前方).
```

```
check(end([文節, 文頭]), 傷前方).
```

```
check(end([文節, 文頭]), 炒め前方).
```

```
check(end([文節, 文頭]), 板目前方).
```

```
check(痛後方, め前方).
```

```
check(傷後方, め前方).
```

```
check(め後方, る前方).
```

```
check(炒め後方, る前方).
```

```
check(る後方, な前方).
```

8. Appendix B

Example Sentences to evaluate the speed of the analysis.

- (1) 人間が、この地球の上で生き続けていくためには、どうしても、自然の恵みに頼らなければならない。
- (2) 私たちが毎日食べる物も、住んでいる家も、着る服も、もとはといえば、みな自然界から手に入れた物である。
- (3) 人間は、その優れた技術を使って、自然から得た物を巧みに加工し、自分たちの生活を豊かにしている。
- (4) 人間にとって、自然は、限らない資源の宝庫なのだ。
- (5) また、人間は、宅地を作るために、山を切り崩して平地にしたり、交通を便利にするために、森を切り開いて道路を作ったりしている。
- (6) あるいは、電気を起こすために、川の流れをせき止めてダムを建設したり、工業地帯にするために、海を埋め立てて陸地に変えたりしている。
- (7) つまり、人間は、いろいろな方法で自然に手を加えているのである。
- (8) このように、人間は、自然が生み出す物を資源として利用する一方、自分たちに都合の良いように自然の姿を変えて生活している。
- (9) この地球上で、人間だけが、自然の資源を思う存分利用したり、自然を改造したりする知恵と力とを備えた生物なのである。
- (10) しかし、それだからといって、人間が思いのままに自然の姿を変え、その資源を手当たり次第に自分たちの物にしてしまってもいいのだろうか。
- (11) ここで、考えてみなければならないのは、自然界には、さまざまな種類の生物たちが、それぞれの環境に応じて生きているということである。

- (12) そして、これらの生物たちは、互いに影響を与え合い、複雑に絡み合った関係を保ちながら、生活しているということである。
- (13) 森という環境を例にとってみよう。
- (14) 森には、いろいろな動物が住んでいる。
- (15) 獣も鳥も虫も、もっと小さな微生物もいる。
- (16) 彼らは、そこに森があるから生活しているといっている。
- (17) 森の植物は、動物たちに食物を提供している。
- (18) 昆虫たちは、木や草の葉を食べたり、花の蜜や、木の幹から出る樹液を吸ったりして生活している。
- (19) 小鳥たちのあるものは、木の実や新芽を盛んについばむ。
- (20) シカやサルは、木の葉や実を好んで餌にする。
- (21) また、動物同士の関係を見ても、昆虫を食べる鳥がいるし、鳥を付け狙うイタチのような獣もいる。
- (22) 虫を餌にしているカエルは、ヘビに一飲み込まれてしまう。
- (23) そのヘビも、タカのような大きな鳥にはかなわない。
- (24) 動物たちが死んだり、落ち葉が積もって土に混じったりすると、土の中の微生物が働き始める。
- (25) カビとかバクテリアのような微生物は、動物や植物の死骸を腐らせ、分解してしまう。
- (26) 分解の結果できた物は、やがて、植物の養分として根から吸い取られる。
- (27) そうして、植物は成長し、動物たちに、再び新しい食物を提供することになるのだ。
- (28) こうして見てくると、森の生物は、互いに食べたり食べられたりしながら、全体としては、うまい具合に釣り合いを保っていることが分かる。
- (29) 彼らは、それぞれでたがひの生き方をしているのではなく、生物同士が、いわば、目に見えない鎖のようなものでつながり合っているともいえる。
- (30) だから、もし何かの原因でこの釣り合いが破れ、生物同士を結んでいた鎖が断ち切られると、思いがけない出来事が起こることもある。

9. Appendix C

Source Grammar to translate into the SAX code in DCG. Numbers in *Italic* face are “identifiers” which are inserted for the purpose of the explanation of the translation.

sentence	→	np, <i>id1</i> vp.
np	→	pron.
np	→	det, <i>id2</i> noun.
pron	→	[I]
vp	→	v, <i>id3</i> np.
det	→	[a].
noun	→	[pen].

```
:- public      sentence/4, np/4, vp/4, pron/4, det/4, noun/4, I/4, v/4, a/4,
               pen/4, fin/1.

np(X,LC) -->   np_zz1(X,LC), np_zz2(X,LC).
sentence([],LC) --> !.
sentence([Id | Tail],LC) -->   sentence_h(Id,LC),!,
                               sentence(Tail,LC).

sentence([_ | T],LC) --> !,    sentence(T,LC).
sentence_h(begin(X,LC),LCC) -->
    [end(X,[LCC | LC]),!].
np_zz1(X,LC) -->
```



```

{tp_ck(X,sentence,New_X2)},
tp_out(New_X2,[id1(New_X2,LC)|T1],T1),!.

np_zz2([],LC) --> !.
np_zz2([Id|Tail],LC) -->
    np_h(Id,LC),!,
    np_zz2(Tail,LC).
np_zz2([_|T],LC) -->
    !,np_zz2(T,LC).
np_h(id3(X,LC),LCC) -->
    vp(X,[vp,{true},LCC|LC]),!.
vp([],LC) --> !.
vp([Id|Tail],LC) -->
    vp_h(Id,LC),!,
    vp(Tail,LC).
vp([_|T],LC) --> !,
    vp(T,LC).
vp_h(id1(X,LC),LCC) -->
    sentence(X,[sentence,{true},LCC|LC]),!.
pron(X,LC) -->
    {tp_ck(X,np,New_X2)},
    np(New_X2,[np,{true},LC]),!.
det(X,LC) -->
    {tp_ck(X,np,New_X2)},
    tp_out(New_X2,[id2(New_X2,LC)|T1],T1),!.
noun([],LC) --> !.
noun([Id|Tail],LC) -->
    noun_h(Id,LC),!,
    noun(Tail,LC).
noun([_|T],LC) --> !,
    noun(T,LC).

```

```

noun_h(id2(X,LC),LCC) -->
    np(X,[np,{true},LCC | LC]),!.
I(X,LC) -->
    {tp_ck(X,pron,New_X2)},
    pron(New_X2,[pron,{true},LC]),!.

v(X,LC) -->
    {tp_ck(X,vp,New_X2)},
    tp_out(New_X2,[id3(New_X2,LC) | T1],T1),!.
a(X,LC) -->
    {tp_ck(X,det,New_X2)},
    det(New_X2,[det,{true},LC]),!.
pen(X,LC) -->
    {tp_ck(X,noun,New_X2)},
    noun(New_X2,[noun,{true},LC]),!.

/**** Link ****/
link(X,X).
link(np,X):-np(X).
link(pron,X):-pron(X).
link(det,X):-det(X).
link(v,X):-v(X).
np(sentence).
pron(np).
pron(sentence).
det(np).
det(sentence).
v(vp).

/**** Id_Table ****/
id_pair(id1,vp).
id_pair(id2,noun).

```


id_pair(id3,np).

id_pair(begin,sentence).

