PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1992,
edited by ICOT. © ICOT, 1992

On A Grammar Formalism, Knowledge Bases and Tools for Natural Language Processing in Logic Programming

SANO, Hiroshi and FUKUMOTO, Fumiyo
Institute for New Generation Technology (ICOT)
Sixth Research Laboratory
sano@icot.or.jp fukumoto@icot.or.jp

Abstract

This paper gives an overview of Natural Language Processing (NLP) by adopting the framework of logic programming in ICOT. First, we introduce a grammar formalism called SFTB, a new grammar formalism which has been evolved from the latest research work on contemporary Japanese. SFTB was designed and developed following the outcome of this research and incorporates computational features. Two grammar studies are in current use at the laboratory. One, Localized Unification Grammar (LUG), is based on the phrase-base approach. Another, Restricted Dependency Grammar (RDG), belongs to the family of dependency grammars. Computer-based dictionaries should be thought of as knowledge bases. We have built a dictionary, in the form of LUG, which is available for sentence processing. In addition to the hand-built database, we have developed computer-based dictionaries. Finally, a tool for developing grammar rules which run on a computer has been introduced. Basic grammar rules, described in the LUG form, have been made using the tool. The tool makes it possible to extend basic grammar rules in order to create adequate grammar rules for user applications. We believe that this set of tools is applicable to a well-integrated NLP system as a whole. Readers who are interested in NLP systems that are not described here should refer to [1].

1 Introduction

In this paper, we describe NLP research activities conducted at ICOTs sixth laboratory. The overall goal is to provide a set of power tools that use NLP and consist of (1) a framework of grammar structures for a language (Japanese), (2) grammar formalisms for writing grammar structures for computational use, (3) non-trivial-sized grammar rules running on a computer, (4) computer-based dictionaries that help build dictionary entries and can be used in grammar rules, (5) tools for analysis sentences, such as a syntactic parser, morphological analyzer, grammar writer's workbench, and a dictionary editor. The power tools contain the results of our research activities on NLP through the underlying logic programming and may be thought of as a well-integrated NLP system.

One of the major problems in the area of NLP is an essential lack of cooperation by the power tools with each other and subsequent shared data, tools and systems. Because of the wide variety of (1) grammar formalisms, (2) parsing mechanisms which are independent developments and (3) forms in which dictionary entries are written, most researchers develop parsing systems individually, write several grammar rules and construct dictionaries as they go. If the many tools for computational linguistics described above can be made available in common, we will be able to make progress through data sharing. To develop a well-integrated NLP system, we have conducted the following research.

A Grammar Formalism

First, we should clarify what we mean by *sentence*. To do so we introduce a grammar formalism which provides a criterion for defining a relationship between the meaning of sentences and a sequence of words. The grammar formalism described here is called SFTB. The underlying framework draws inspiration from Japanese language morphology [10] and from the latest research work on contemporary Japanese such as Japanese Phrase Structure Grammar (JPSG)[3]. The framework is intended for use in computational grammar.

Two Grammar Descriptive Frameworks

The next objective is to investigate a grammar descriptive framework to hand-code grammar rules as linguistic information supplied by means of SFTB. These rules should be applicable to computer processing in the framework of logic programming. There are two grammar structures in current use at the laboratory. One, Localized Unification Grammar (LUG), is based on the phrase-base approach and aims at unification based formalisms. Another, Restricted Dependency Grammar (RDG), belongs to the family of dependency grammars and processes sentences in a traditional way.

Dictionaries

We shall describe three kinds of dictionaries. Lexical information used in a computer was required in an implementation of LUG's grammar. This is characterized as the finite syntactic information of attribute value pairs and results in a hand-built dictionary. Although the computer resident dictionary, consisting of about 7,000 entries, is hand-coded, the lexical database for morphological analysis created from existing computer-based resources by a conversion program consists of 150,000 entries. We have a dictionary where each entry has a large amount of syntactic information and sense-related semantic information. It may be thought of as a linguistic knowledge base.

A Tool

Finally, we introduce a grammar rule development system called LINGUIST. LINGUIST consists of a bottom-up parser (BUP-translator[8]) and debugging facilities with a cooperative response coordinator for user interfaces. The system is being integrated into an environment of support tools for developing, testing, and debugging grammar rules written in LUG. With this system, we have developed the basic grammar, which has 800 grammar rules, for contemporary Japanese language including morphological analysis rules.

For academic use, tapes are obtainable free from ICOT. These tapes serve as the linguistic tools mentioned above.

2 SFTB – A Grammar formalism for the Japanese language

The aim of developing SFTB is to investigate linguistic frameworks for computational processing of the Japanese language. This framework is a necessary grammatical basis for processing Japanese sentences by computer. A grammatical basis for a language should provide a concrete and coherent framework for relating the linguistic form and the content conveyed by sentence expression. A computer must operate on the information structure expressing the content of a sentence produced by the framework.

In the SFTB framework, the syntactic structure derived from the linguistic form is based on a compositional line of sentence analysis, but not on a flat dependency analysis, which is the traditional way of handling sentence structure. It is able to cope with the problems of subject and object omission, ellipsis, interdependence on context and so forth. It may also end structureless and patternless Japanese language confusion caused by the lack of a proper syntactic framework.

2.1 The SFTB grammar system

In our grammar system, the central units will be that of morphemes but not words, which are classified into parts of speech generally. The grammar proposed in this paper has morphemics as a part of its grammar system. This is a grammar system which is rooted in [10] (see the survey).

Morphemes come in several varieties. A basic unit called *goki* stands for a concept where a certain relation holds the state of affairs and the idea that the object belongs in the real (cognitive) universe. Units of this kind can be divided into two types. One of the free forms, *jiritsu-goki* forms words by itself. We call one of the bound forms *ketugou-goki*, since the unit must combine with affix(s) in order to form words.

A unit called *setsuji* performs a grammatical function where states of affairs are linked with certain relations. For example, verb endings are a type of morpheme. This approach uses a neutral unit *goki* in relation to the grammatical behavior under syntax.

Example 1 shows a phrase structure produced SFTB framework for the phrase "*Uresi sa no taiigen*" ("Expression of happiness" in English). In this case, the word *Uresi-sa* (Happiness) is derived from the *Uresi*(*Keiyou-goki*) and the affix *sa*.

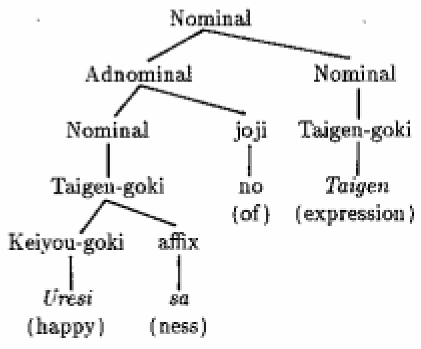

Figure 1: Example of phrase structure

Note that, although the phrase structure is a constituent structure representation from a structural point of view, nodes including feature information are more complex i.e. feature bundles instead of category name. Semantic information is also added to the feature bundles. To make this approach applicable to sentence analysis, all that needs to be done is to integrate morphology and syntax into a comprehensive system.

2.2 Basic patterns and sentence structure

The integration required reconsideration of not only the relation between morphology and syntax but also the general framework of the Japanese language, such as conjugation lists of verbs, word classification , basic sentence patterns and so forth. In this section, we concentrate on the grammar basis for syntactic analysis for Japanese language applicable to computer processing.

First, we should clarify what we mean by a sentence as a criterion. No sentence is uttered or written without the speaker or writer expressing their view on a subject. This may suggest that we ought to extract not only the contents of the sentence but also the intention of the originator from the surface structure of the sentence. Above all, the syntactic forms mapped to verb endings are to be closely related to the meaning of the sentence.

However, this is an ideal case where actual usage depends on context and discourse. As you know, speech is often rambling. One may ask how the content and intuition of graffiti can be extracted. Of course, the criterion mentioned above has to be applied to a limited range of linguistic phenomena. While bearing the above in mind, we offer the criteria and seek sentence structure to map the surface string to an internal representation that is used for NLP by computers.

We characterize the properties of sentence structures we have been investigating, as illustrated in Figure 2.

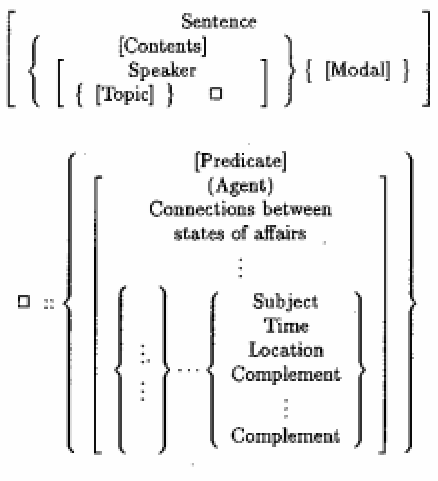

Figure 2: The Sentence Structure

The basic patterns of Japanese sentences are presented in Table 1. The verb "suru", to do, is taken up as a model.

Several types of conjugation lists for verbs are available and the elements of verb endings will be tailored to meet the sentence endings of basic patterns. The lists consist of sentence endings and phrase endings appearing in loosely dependent clauses. The derivation of the conjugation lists illustrated in Table 2 are based on the assumption that verb endings are proportional to the linguistic clues depending on the meaning that the sentence conveys.

Table 1: Basic patterns

Pattern	Sentence endings	Intention
Indicative	*suru/sita*	Neutral
Presumptive	*surudarou/sitadarou*	Presumptive
Volitional	*siyou/sitadarou*	Volitional
Imperative	*siro/suruna*	Imperative

Table 2: Conjugation lists

	Lists (SFTB)		Lists †	
	Form	Type	*Form*	Type
Hanasu	*su*	Non-perfect indicative	*sa*	1
(To speak)	*sita*	Perfect	*so*	2
	sudaraou	Non-perfect presumptive	*si*	3
	sitadarou	Perfect presumptive	*su*	4
	sou	Positive volitional	*su*	5
	sumai	Negative volitional	*se*	6
	se	Positive imperative	*se*	7
	suna	Negative imperative		
	si	Connective form 1		
	site	Connective form 2		
	seba	Non-perfect conditional		
	sitara	Perfect conditional		
	sitari	Coordinate form		

† School grammar : 1 : Mizen-kei, 2 : Mizen-kei, 3 : Renyou-kei, 4 : Shuusi-kei, 5 : Rentai-kei, 6 : Katei-kei, 7 : Meirei-kei

Compare SFTB's conjugation lists with the grammar lists of the school. The number of verb endings in the school grammar lists is less than seven, in the SFTB's lists, there are over a dozen. For each verb ending form, however, a type name provides sentence patterns, a syntactic function link to the meaning expressed by the sentence.

3 Linguistic Knowledge Bases – Grammar

The well understood way of processing Japanese language is that you read the technical papers and understand the most important part of the framework. Since linguistic knowledge about grammar and dictionaries is described in the natural language itself, there is no straightforward means of applying linguistic knowledge investigated by SFTB to map underlying descriptions onto grammar rules that run on computers. The weakness is in the representations of the programs underlying the parser available for language processing on computers. To solve the problem, we present two computational grammar descriptive frameworks for developing grammar rules built from the SFTB framework. For computer systems with the parser developed in the logic programming framework, these grammar structures give the grammar writer a descriptive framework for writing grammar rules to be used by a parser.

In this section, we describe two computational grammar descriptive frameworks, one is LUG formalism, the other is RDG formalism (RDG), used for writing grammar rules that run on computers. The former is based on the unification grammar that belongs to the phrase structure base grammar. The formalism of the latter takes its stand on the dependency structure grammar.

Much has been done in writing grammar rules to parse natural languages. These grammar rules generally make use of an assumed grammar formalism. Almost no attempt, however, has been made to utilize different ways that base their underlying formalisms on the same grammar for processing a natural language. Thus, although processing methods have been discussed that contrast parsing speeds, required memories and so forth, we have not talked about the merits and the demerits of these grammar structures. In order to ascertain what kind of grammar formalism is appropriate for processing Japanese language, different approaches must be applied to the language.

The original goal of SFTB was to provide a new Japanese language grammar formalism for contemporary Japanese. As applicability to computational linguistics look possible, we are now concentrating on writing grammar rules in terms of LUG and RDG, in the framework of SFTB.

3.1 A Phrase Based Approach – LUG

Definite Clause Grammar (DCG)[6] is one of the bridges connecting NLP and logic programming. Most of our grammar research activities can be regarded as improvements and extensions of DCG. A wide range of parsing techniquea have been suggested based on DCG through underlying context-free grammar. Although the computational effectiveness of DCG is powerful enough to write grammar rules that run directly on a computer, it can be thought of as programming rather than the description of the grammar of a language. It may be said that DCG is less expressive than other grammar formalisms in the sense of mathematical measures. If the process of developing grammar rules is tied to the description of DCG, it will be difficult to develop large grammar structures manual by using the DCG description. To overcome this problem, we have designed LUG to be accessible to grammar writers with little computer experience. Thus, LUG is a grammar specification language designed for users to develop non-trivial grammar expressed in the DCG.

The basic data of LUG is a feature syntax. Categories are expressed as feature sets. Since the feature sets are represented as Prolog lists, the grammar is written in DCG formalisms, allowing users to make use of the BUP[8], BUP-XG[12], SAX[9] translators being developed in the framework of logic programming. As a sample LUG, we present, in Figure 3, an informal representation of the phrase "*Uresi sa no taigen* .

$$\begin{bmatrix} \text{Morph} & \textit{taigen} \\ \text{Category} & \text{Taigen} \\ \text{Marker} & \textit{no} \\ \text{oftype} & \begin{bmatrix} \text{Morph} & \textit{uresi/sa} \\ \text{Category} & \text{Taigen} \\ \text{Marker} & \textit{sa} \\ \text{Sem} & \text{nominalize}(Y) \end{bmatrix} \\ \text{Sem} & \text{oftype}(X,\text{nominalize}(Y)) \end{bmatrix}$$

Figure 3: LUG for the phrase

The form produced by LUG can be described as a complex constituent that is the result of compositional and functional application. The functional application is used to limit compositional ambiguities caused by unification-oriented structural description. The contextual information of knowledge bases is dealing with the world or pragmatic knowledge about words, for example can be re-unified later. Thus, complete resolution of constituent structures depends on semantic-based and pragmatic-based accounts of subsequent information. With this formalism, the Japanese language grammar is written independently of the task of the application domain.

3.1.1 The Basic Grammars

The LUG formalism has been used to build grammar structures for basic coverage of contemporary Japanese. As of now, grammar structures of 800 grammar rules are usable and are under development for the purpose of increasing coverage.

Remember, however, that the readability of grammar structures is sacrificed when its rules are extended. It is often difficult to keep a large number of grammar rules under control. Even a small loss of attention causing inconsistent grammar can cause ambiguities to increase and analysis to become useless. An important characteristic of the basic grammar structure is that it is orderly divided into 12 groups to the following standards:

○ Difficulty in analyzing sentences

According to Figure 2, a complete analysis of sentences comes from success in understanding the syntactic elements in the structure when in the parsing process. This is directly and indirectly related to the basic sentence patterns and the sequence that words appear in sentences. The fewer syntactic elements omitted, the easier it is to parse its structure. The greater the difference between the syntactic elements and their corresponding morphemes, the more it costs to analyze a sentence correctly and to grasp its meaning.

Grammar structures can be loosely divided into three levels: elementary, intermediate and advanced levels. We list here some samples of the kinds of grammar structure levels, but are limited to the following.

Elementary level

decision(declaratives), supposition, conjectural form(declaratives), command(imperative), aspect operators, negation, polite form, complements, mood auxiliaries.

Intermediate level

passives, causatives, modal adverbs, spacio-temporal adverbs, topicalized phrases, relatives.

Advanced level

conditional phrases, causal phrases, some connectives, conjunctions and disjunctions of nominal phrases.

3.2 A Dependency Based Approach – RDG

Japanese word order is said to be free. Thus, dependency grammar that only focuses on the relation between two arbitrary constituents as a syntactic structure in a sentence has been well studied. Many NLP systems for the Japanese language have adopted the dependency paradigm as an approach for syntactic analysis.

However, the problem of the dependency structure (it is not a tree but a connected graph structure) which is used in these NLP systems is that useless solutions are generated, which bring about a combinatorial explosion. This is because whether one constituent of a sentence modifies another constituent concerns only the localized information between the two constituents, such as the selectional restriction between a verb and its complements.

In this section, we propose a dependency grammar formalism for the Japanese language called Restricted Dependency Grammar (RDG). A characteristic of RDG is (1) The interpretation of whether one constituent modifies the other or not depends on global information based on the word order of a sentence. So we can suppress the generation of useless solutions. (2) Every constituent of a sentence except the last should modify at least one constituent on its right. So, some linguistic phenomena, themes or ellipses can be treated easily in our approach.

RDG is currently implemented in the SICStus prolog, and is being evaluated by using a Japanese newspaper editorial, with specially attention given to the number of solutions.

In the following subsections, we introduce an outline of RDG formalism, concentrating on the constraint based on the word order of a sentence.

3.2.1 Modifiability rank

A sentence consists of many constituents. We call these phrases. Every phrase of a sentence has two syntactic contrastive aspects. that is, one phrase modifies the other and one phrase is modified by the other. We call these aspects the modifier (henceforth Mer) and the modificand (Mcand). When the Mer of one phrase and the Mcand of another phrase match, we can connect the two phrases by an arc. In RDG formalism, every phrase and arc have a modifiability rank value.

The phrase rank is a classification of a phrase based on the number of Mer and Mcand phrases. For example, a manner adverb such as "yukkuri" (slowly) can modify verbs such as "yomi-nagara" (reading a book), "yomeba" (if you are reading), "yomu-node" (as you read), and "yonda-keredo" (though you read). On the other hand, of all these verbs, a modal adverb, such as "tabun" (probably), can only modify "yonda-keredo" (though you read). This means that the number of Mer "yukkuri" (slowly) is more than that of "tabun" (probably). In a similar way, if a phrase can be modified more than another phrase, the number of Mcand phrases is more than that of the other phrase.

The phrase rank consists of 7 Mer and 7 Mcand ranks. Every phrase has a Mer and Mcand rank. The classification of phrases based on these ranks is given in Table 3.

Table 3: Classification of phrases based on rank

Mer	phrase example
a_1	deictic pronoun
a_2	manner adverb
a_3	noun
a_4	continuous verb
b	condition verb. temporal adverb
c	causal verb
d	contrastive verb, modal adverb

Mcand	phrase example
$\mathbf{A}_1$	deictic pronoun
$\mathbf{A}_2$	manner adverb
$\mathbf{A}_3$	noun
$\mathbf{A}_4$	continuous verb
B	condition verb
C	causal verb
D	contrastive verb

Conditions between phrase ranks are formulated as in (1), and (2).

$$a_2 \succ a_3 \succ a_4 \succ b \succ c \succ d \qquad (1)$$

$$a_2 \prec a_3 \prec a_4 \prec \mathbf{B} \prec \mathbf{C} \prec \mathbf{D} \qquad (2)$$

(1) shows the Mer rank. (2) shows the Mcand rank. For example, when a manner adverb "yukkuri" (slowly) is classified as **a$_2$**, and a modal adverb "tabun" (probably) is classified as **d** from Table 3 Mer Rank, we found that the number of Mer "yukkuri" (slowly) is more than "tabun" (probably) from formula (1).

The arc rank is a classification of an arc based on the phrase's modifiability rank. It is incorporated in the word order of a sentence. We assume that a sentence consists of three phrases, P_i, P_j, and P_k ($i < j < k$). We can get the dependency structures (Fig 4 (a), and (b)) from this sequence. The arc between P_i and P_j is shown as $\overrightarrow{P_iP_j}$. In Fig 4 (a), we call $\overrightarrow{P_iP_j}$ an adjacent arc of $\overrightarrow{P_jP_k}$. In Fig 4 (b), we call $\overrightarrow{P_jP_k}$ the inside arc of $\overrightarrow{P_iP_k}$.

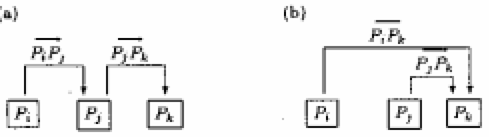
(a)
$\overrightarrow{P_iP_j}$
$\overrightarrow{P_jP_k}$
P_i
P_j
P_k
(b)
$\overrightarrow{P_iP_k}$
$\overrightarrow{P_jP_k}$
P_i
P_j
P_k

Figure 4: Dependency structure

(1) [Kare-ga] yobu-to heya-kara dete-kita.
When he called ϕ, ϕ went out of the room.
(1)' Yobu-to [kare-ga] heya-kara dete-kita.
When ϕ called him, he went out of the room.

(1)' shows a (1) [kare-ga (he)] /[yobu-to (when called)] conversion. We found that the meaning of (1) is different from that of (1)'. When we read sentence (1), we pause on the phrase "yobu-to" (when called). That is, the temporal particle "to" has a function that temporarily disconnects the sentence. So it is hard to say that "kare-ga" (he), which exists before "yobu-to" (when called), is able to modify "dete-kita" (went out of) across a phrase "yobu-to" (when called). This means that there exists a word order constraint between the phrase "kare-ga" (he) and "yobu-to" (when called). Thus, we can get one of the solutions shown in Fig 4 (a). In (1), "Kare-ga" (he) equals P_i, "yobu-to" (when called) equals P_j, and "dete-kita" (went out of) equals P_k.

The rank of an arc incorporates these phenomena (the word order of a sentence). The rank of an arc consists of four levels, corresponding to a phrase's function, that temporarily disconnect the sentence. These four levels are represented by **a**, **b**, **c**, and **d**. These values depend on the values of the Mer and the Mcand rank shown in Table 3. Rank of phrase P_i, P_j, and an arc $\overrightarrow{P_iP_j}$ are shown in Table 4.
In Table 4, the Mer rank of P_i is on the left and the Mcand rank of phrase P_j is on the top. The column shows the rank of an arc $\overrightarrow{P_iP_j}$. A blank column indicates that there is no arc between these two phrases. Conditions between the rank of two arc's is formulated in (3).

Table 4: Rank of P_i, P_j and $\overrightarrow{P_iP_j}$

	A$_1$	**A$_2$**	**A$_3$**	**A$_4$**	**B**	**C**	**D**
a$_1$	a						
a$_2$		a	a	a	a	a	a
a$_3$			a	a	a	a	a
a$_4$				a	a	a	a
b					b	b	b
c						c	c
d							d

In (3), for example, when $\mathbf{a} \succ \mathbf{b}$ we say that **b** is lower than **a**.

$$\mathbf{a} \succ \mathbf{b} \succ \mathbf{c} \succ \mathbf{d} \tag{3}$$

Now we show the constraints of word order using the rank of an arc. When the dependency structure is as in Figure 4 (a), the rank between the two arcs should be satisfied by formula (4). When the dependency structure is as in Figure 4 (b), the rank between the two arcs should be satisfied by formula (5).

$$\text{Rank of } \overrightarrow{P_iP_j} \;\succeq\; \text{Rank of } \overrightarrow{P_jP_k} \tag{4}$$

$$\text{Rank of } \overrightarrow{P_jP_k} \;\succeq\; \text{Rank of } \overrightarrow{P_iP_k} \tag{5}$$

3.2.2 The RDG formalism

In RDG formalism, there are two different kinds of constraint. One is how to make an arc which connects a pair of phrases. The other is whether we can make an arc. These constraints are described as follows.

Structural constraints

1. Absolute dependency
   Every phrase of a sentence except the last should modify at least one phrase on its right. If a phrase modifies a phrase on its right, we can connect them with an arc. The last phrase modifies no other phrase.

2. Crossing
   No two arcs should cross each other.

Linguistic constraints

1. Constraints for phrases
   When P_i and P_j satisfy Table 4, we can get the dependency relation between P_i and P_j. Its column value is the rank value of an arc.

2. Constraints for arcs
   When P_i modifies P_j, (i) the rank of $\overrightarrow{P_iP_j}$ should be lower than the rank of its adjacent arc, (ii) the rank

of $\overrightarrow{P_iP_j}$ should be lower than the rank of its inside arc.

4 Linguistic Knowledge Bases – Dictionaries

Terms used to refer to these dictionaries are (1) computer resident dictionary to be used in syntactic processing by the parser, (2) computer-based dictionary being applicable to applications such as morphological analysis, and (3) machine-tractable dictionary containing lexical information to be interpreted by human readers. i.e. a database used in a data base management system.

The lexical data of these computerized dictionaries includes the following information in (3) only: a list of possible words in a given language (in our research, that language was Japanese), a list of base words and their inflected and derived forms. A classification of semantic information such as word senses.

For dictionary (1), since the lexical data is not necessarily stored in one place, syntactic information about category and the subcategorization behavior of words will appear in grammar rules implicitly.

4.1 Dictionary available in LUG

The dictionary was built by analyzing data from a standard elementary school text. The computer resident dictionary consists of 7,000 entries for each entry in LUG form. Work on the application of the dictionary to the grammar structure has focused on the development of grammar structures written in LUG. Hence, the dictionary relies on our hand-built database and tends to be rather limited in size.

4.2 Dictionary used for Applications

Taking limitations into account, we have developed a program which provides a way of constructing lexical databases through the available machine-readable dictionaries. The lexical database, consisting of 150.000 entries, was created from existing machine-readable dictionaries by using the program. We intend to provide the dictionary as a resource for morphological analysis.

4.3 Dictionary available in DB

A compact but high-capacity computer resident dictionary was extracted from machine-readable dictionaries supplied by IPA [4, 5] by a specialized program. The dictionary can be thought as a linguistic knowledge base and can assist in constructing a restricted specific-domain dictionary for used in NLP. by providing semantic information to the analysis. At present, we utilize the dictionary by using a data base management system.

5 Tools for NLP

We have formalized the SFTB grammar formalism for the Japanese language and grammar rules running on computers are realized in the framework of logic programming using LUG and RDG to demonstrate the descriptive power of their grammar formalisms. On the other hand, we have developed NLP systems for doing computational linguistics with logic programming techniques, such as a dictionary management system, a grammar writer's workbench, a debugging system built around a parser and so forth. In the following section, we introduce a grammar rules development system.

5.1 LINGUIST

LINGUIST is a NLP system with three purposes:

(1) verifying the more detailed nature of the framework of a natural language (Japanese) in a strict enough sense to take an objective view;

(2) developing more useful grammar structures that can be used widely in the domains where the natural language interface to an information retrieval component is used as an intelligent system device: and

(3) having a tool for processing Japanese language and thoroughly trying our grammar ideas.

In the sense of (1), LINGUIST is a grammar development system designed to assist the development of grammar rules expressed in the DCG formalism. LUG, mentioned in Section 3.1 is currently implemented in the LINGUIST, and has been doing well as a development and verification tool in the research of Japanese grammar with respect to computational linguistics. The primary goal in designing the LINGUIST was to efficiently develop grammar rules by computer using logic programming. Thus, the system described with respect to (1) functions as a grammar writer's workbench for NLP.

Once produced, the grammar rules are included in the NLP unit that makes up one part of the user interface of a system, such as an expert system, and the translator of a machine translation system. Then the grammar rules are adjusted to a particular purpose for which the parser applying the grammar to sentence analysis must be able to produce the structure needed by the application domain. Modification of the grammar rules increases with greater application speciality. When this happens, it becomes unclear what the basic grammar rules are.

In order to avoid this problem, LINGUIST has been enhanced with a powerful editing facility that makes it easy to support modification of the grammar rules needed for adjustment to an application domain. With this improvement, the LINGUIST provides users with the basic grammar rules being developed at ICOT. The application of the basic grammar rules within the LINGUIST simplifies many of the modifications dealt with by the application system builders, as mentioned in (2).

In (3), the system being developed in the framework of logic programming will enable the investigation and processing of various linguistic phenomena in the approach of parsing a natural language (Japanese) by computer.

5.1.1 The Machinery of the LINGUIST

This system initially implemented on PSI-II[1] under ESP[2] is now available on a SPARCstation under CESP[3].

The LINGUIST is organized into three major modules, as illustrated in Figure 5:

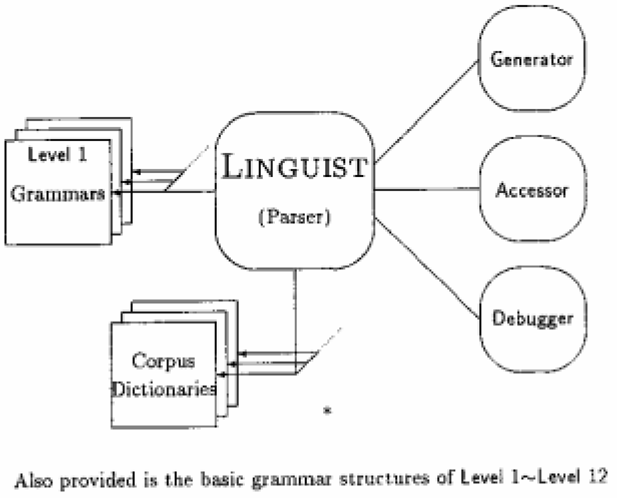

Figure 5: The System Configuration

○ **Generator** includes the BUP translator that translates Grammar rules written in DCG into ESP (CESP) code. The user may manually select operations to consult with, save or load a parser program.

○ **Accessor** allows the user to access the parser that is ready to parse and provides inspectors to display the results of parsing. Each inspector provides an iconic menu of operations. Grammar structures can be tested interactively and the results saved on system holders for inspect and on files for future reference.

[1]Personal Sequential Inference Machine developed at ICOT.
[2]Extended Self-contained Prolog developed at ICOT.
[3]Common ESP based on ESP developed at AI Language Institute Corp.

○ **Debugger** provides a debugging tool for visualizing each invocation during parsing as mouse-sensitive operations. A tracing facility that follows the progress left by the parser is built into the module.

Together these provide a menu-based interface that makes communication with the LINGUIST easy.

5.2 Other applications of the system

As of autumn 1991, the LINGUIST consists of the software itself, a set of grammar rules and a set of dictionaries used by it. The LINGUIST allows an advanced user to extend and modify the basic grammar rules for a specific usage. On the other hand, for inexperienced users the system with its debugging facilities allows the process to be mentioned and helps comprehension of how the grammar rules are applied.

6 Final Remarks

In this paper, we presented an overview of the achievements of our lengthy study in the ICOT project: (1) A framework of Japanese grammar called SFTB, (2) Two grammar formalisms, LUG and RDG, both based on logic programming, (3) Dictionaries thought of as linguistic databases, (4) A grammar rule development system, LINGUIST.

In point (1), our framework of Japanese grammar has the sentence level and structural aspects of Japanese sentence construction. It, also, covers the basic linguistic phenomena of contemporary Japanese and these phenomena are systematically ordered in accordance with the standards, as mentioned in Section 3.1.1.

Section 3 of the paper outlined two grammar formalisms. LUG is a unification-based grammar formalism whose syntactic notation is DCG and is a kind of context-free structure grammar. At present, grammar rules written in LUG formalism on the basis of SFTB are under evaluation by using elementary school texts. On the other hand, RDG is the based on dependency grammar formalism with a mechanism producing the connected graph structures of sentences. We are currently testing its descriptive power by using Japanese newspaper editorials.

In Section 4, we described several kinds of dictionaries thought of as linguistic databases. First, the dictionary consists of about 7,000 entries and is used for analysis as part of the LUG grammar structure. Each entry has the detailed information needed to perform morphological analysis and syntactic parsing. Second, the TRIE structure dictionary has a huge number of dictionary entries built for use in morphological analysis. The dictionary consists of about 150,000 entries.

In Section 5, a tool for the grammar rules development system called LINGUIST was introduced. With

LINGUIST we developed basic grammar structures with 800 grammar rules written in LUG formalism. Thus. LINGUIST provides not only an environment for the development of grammar rules but also an environment for modifying grammar rules that will be utilized in various NLP application systems.

The LINGUIST software and the basic grammar rules mentioned above are available to the general public. The dictionaries are also freeing available. We hope these are extensively utilized in various NLP systems.

Availability

Academic users of LINGUIST can obtain free a magnetic tape of the CESP code of the LINGUIST system. The basic grammar rules for the Japanese language in LUG is available as an appendix to the LINGUIST system. The software can be ordered from Mr.Yukio Shigihara, Deputy Chief Research Planning Section, Research Planning Department Research Center, ICOT.

References

[1] Akasaka. K., *et al.* (1989). Language Tool Box (LTB) A Program Library of NLP Tools. *ICOT Technical Report: TR-521. ICOT.*

[2] Fukumoto, Fumiyo., *et al.* (1991). A Framework for Restricted Dependency Grammar. *Papers from 3rd International Workshop on Natural Language Understanding and Logic Programming, pages 68–81.*

[3] Gunji, Takao (1987). Japanese Phrase Structure Grammar A unification-Based Approach. *Studies in Natural Language and Linguistic Theory, D.Reidel Publishing Company.*

[4] Keisankiyou Nihongo Kihon Doushi Jisho IPAL (Basic Verbs) (in Japanese), (1987). *Information Technology Promotion Agency, Japan.*

[5] Keisankiyou Nihongo Kihon Keiyoushi Jisho IPAL (Basic Adjectives) (in Japanese). (1990). *Information Technology Promotion Agency. Japan.*

[6] Pereira, F. & Warren, D (1980). Definite clause grammar for language analysis - a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence. Vol.13. No.3. pages 231-278.*

[7] Masuoka, Takashi & Takubo. Yukinori (1989). Kiso Nihongo Bunpou (in Japanese). Tokyo. Kuroshio Shuppan.

[8] Matsumoto, Yuji., *et al.* (1983). BUP : A Bottom-Up Parser Embedded in Prolog. *New Generation Computing, Vol.1. No.2. pages 145–158.*

[9] Matsumoto, Yuji. & Sugimura, Ryouichi (1986). Ronrigata Gengoni motozuku Koubun Kaiseki Sisutemu SAX (In Japanese), *Computer Software. Vol.3. No.2. Japan Society for Software Science and Technology.*

[10] Morioka. Kenji (1987). Goi no Kousei (in Japanese). Tokyo, Meiji Shoin.

[11] Sano. Hiroshi (1990a) Shizen Gengo Jikken Shien Kankyou LINGUIST (in Japanese), *Papers form 8th Symposium on Fifth Generation Computer Technology, ICOT.*

[12] Tokunaga, Takenobu., *et al.* (1988). LangLAB : A Natural Language Analysis System, *Papers from 12th International Conference on Computational Linguistics, Vol.II.*