J-Analyser; Analyser of Japanese Sentences Based on Binding Structure Representation of Words

TAKASHI IKEDA*

An analysis system of the Japanese sentences (J-Analyser) is presented. J-Analyser reads Japanese sentence written in Romanized Japanese and builds up its semantic structure. The semantic structure is represented in some nested form called *phrase*.

J-Analyser aims at reflecting faithfully the nature of the Japanese language which is agglutinative and SOV type language. It is grounded upon dependency analysis and stack model. It uses a binding structure representation of a word that is a generalized view of a case frame structure. In this paper, not only usual case structure but several kind of binding rules are presented. There are two types of them, one is an acceptor type rule and another is a donor type rule. They can be connected with not only content words but also function words and phrase categories. Various phraseology can be naturally described by these rules.

J-Analyser is composed of the segmentation section, the reorganization section, the transformation section, the dependency analysis section and the context analysis section.

1. Introduction

This paper describes a system (J-Analyser) that analyses Japanese sentences. The analyser reads Japanese sentences written in Romanized Japanese and builds up its semantic structures. As a practical application of the result of the analysis, a translation system into another language is presented.

Japanese is an agglutinative and SOV type language quite different from European languages. Unlike SVO type languaees a dependent necessarily precedes its governor (i.e. a governor always comes out after its dependents), and the basic grammatical relations are not decided by word order but by postpositions agglutinated to independent words. And further, Japanese sentences usually involve many elliptical constructions.

J-Analyser is designed in due consideration of these characteristics of the Japanese language. It analyses Japanese sentences by the principal use of binding structure representation of words. It does not take an automaton-like mechanical view of language, and is not grounded upon an acceptor machine model. It is grounded upon an idea of comparing a language understanding process to chemical reactions. The following is the analogical image of a language understanding process.

A word is analogous to some chemicals. And understandung sentences is compared to a process of combining these chemicals. The meaning of the sentences is, as it were, the most stable compound among the products.

Describing the nature of chemicals is analogous to describing the meaning and uses of words. The binding structure of words is an analogy of bonds of chemicals.

Solving chemical reaction equation is compared to the analysis of sentences by a computer. The environmental condition of reaction is, as it were, the situation of the utterance, the context, or the world knowledge that the understander has.

2. Representation of Semantic Structure of Sentences

J-Analyser analyses input sentences and rebuilds its semantic structures. The semantic structure is represented here in the following nested form, and we call this a "*phrase*".

$((pc\ fw_1\ fw_2 \cdots cxt)\ cw\ S_1\ S_2 \cdots)$

where $pc \cdots$ Category of the phrase.
$fw_i \cdots$ Function word.
$cxt \cdots$ Context register.
$cw \cdots$ Content word, the nucleus of the phrase.
$S_i \cdots$ Specification of the phrase. S_i is again a phrase or a phrase variable.

Fig. 1 is an example of a phrase representation of a sentence.

A phrase category *pc* is a string of a form *s*: *d*; where *s*: designates a surface syntactical characteristics of a phrase and *:d* designates a deep semantical characteristics of a phrase. For example, in Fig. 1, a phrase whose category is associated with *:a* and a phrase with *:b* are, on their concepts, in belonging relation [*:a*] ∋ [*:b*]. A phrase with *:g* has an objective relation to its governing noun phrase. As for *s*:, a group of function words that syntactically bind a phrase to its governor is decided mainly by *s*:. It indicates a syntactic relation between a phrase and its governor. In Fig. 1, **ga**: indicates that the phrase is the nominative case of its governor and a binding function word is chosen from

*Electrotechnical Laboratory, Ibaraki, Japan.

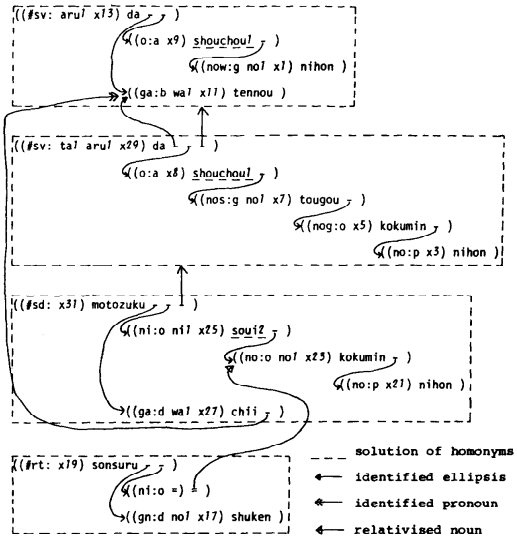


Fig. 1 Phrase representation of Article 1 of the Constitution of Japan. "*tennouwa nihonkokuno shouchoudeari nihon kokumin tougouno shouchou deatte, kono chiiwa shukenno sonsuru nihon kokuminno souini motozuku.*"

(ga, wa, mo,···), **no**: indicates that the phrase is a nominal modifier of its nominal governor and a binding function word is "no", **now**: indicates the same as **no**: except that the phrase with **now**: is a metamorphism of an objective case of a verbal phrase and that its nominal governor is a metamorphism of the verbal phrase, and so forth. The fine categorization of phrases is a problem to be investigated further.

A function word plays a role of binding hand, transforms the basic binding structure of the content word, or indicates tense, mood or some other aspects.

The context register holds ellipses in the phrase (i.e. S_i that are left as phrase variables) or pairs of an ellipsis and its identified phrase, pronouns in the phrase or pairs of a pronoun and its referent phrase, goodness of the phrase, or any other contextual information connected with the phrase. In Fig. 1, x29 holds the identified phrase for the first argument of "da" that is shortened in the surface, x27 holds the identified phrase for the argument of "chii" that is pronominalized in the surface.

A phrase variable describes conditions to be fulfilled by a phrase which should specify the phrase and should be put in the place of the phrase variable itself. To be concrete, a phrase variable is a component of a generalized case frame ($FRULE, B) described in the next section.

A content word *cw* represents a central meaning of the phrase. It governs S_i's as obligatory specifiers. S_i's and *cw* are in dependents-governor relation.

This phrase structure does not represent deep logical semantics. J-Analyser intends to extract rather surface meanings and binding relations between words and phrases.

3. Representation of Binding Structure of Words

Binding structure of a word is dependency and governing properties peculiar to each word or word group. Binding structures are represented in binding rules linked to words or phrase categories.

Binding rules are divided into two types (Fig. 2). One is an acceptor type rule ($FRULE) that describes what kind of phrases a phrase requires or admits as specifications of the phrase. The other is a donor type rule ($MRULE) that describes what kind of phrases a phrase can or should specify. For example, case relations between noun phrases and a predicate are naturally described by an acceptor type rule linked to a predicate, while it is more natural to use a donor type rule to describe a binding condition between an adverb phrase and a noun or a predicate ("motto migi", "motto taberu",···). There are many other binding relations which are easier and more natural to describe by donor type rules than by acceptor type rules. The followings are such examples.

"*denwa wo* kakeru"···(to telephone)
"*denpou wo* utsu"···(to telegram)
"*ikuta no* jiken"···(*many* affairs)
"*Tokyo de* kau"···(to buy *at Tokyo*)
"*yoru* yomu"···(to read *at night*)

The following are the steps of procedure to decide whether two phrases are to be bound together into a single phrase, that is, whether a phrase (mp) can be a specification of another phrase (fp).

$$\text{mp} = ((pc\text{–}m\cdots fw_i\text{–}m\cdots cxt\text{–}m)\, cw\text{–}m\cdots S_i\text{–}m\cdots)$$
$$? \Downarrow$$
$$\text{fp} = ((pc\text{–}f\cdots fw_i\text{–}f\cdots cxt\text{–}f)\, cw\text{–}f\cdots S_i\text{–}f\cdots)$$

(1) Examine $FRULE linked to *cw–f*.
(2) Examine $FRULE linked to *pc–f*.
(3) Examine $FRULE linked to fw_i–*f*.
(4) Examine $MRULE linked to *cw–m*.
(5) Examine $MRULE linked to *pc–m*.
(6) Examine $MRULE linked to fw_i–*m*.

$FRULE's and $MRULE's are simple Lisp programs. To examine these rules is simply to fetch and evaluate the programs. If the result of the evaluation is not empty (i.e. not NIL), the succeeding steps are skipped.

The greater part of binding relations are decided at step (5). Most of $MRULE of *pc*−*m* are conditional calls of declarative type rules ($MRULE, B and $FRULE, B···described later). For example, $MRULE of NOUN and ADVERB are such that;

"If a candidate phrase for governor is predicative, there

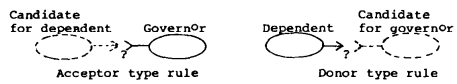


Fig. 2 Two types of binding rules.

is a possibility of binding. Then examine $FRULE,B linked to the predicate, examine $MRULE,B linked to the nominal phrase and then examine a possibility of combinative expression. If a candidate phrase for governor is nominal and a dependent nominal phrase is accompanied by a binding function word "no", then enter the routine for noun sequence with "no" (see [4–4], Fig. 4). Otherwise there is no possibility of binding."

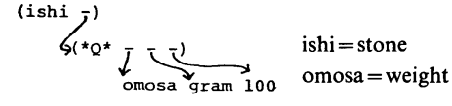
"If a candidate phrase for governor is nominal, adverbial or predicative, there is a possibility of binding. Examine $MRULE,B linked to the adverb. Otherwise there is no possibility of binding."

From step (1) to step (4) and step (6) are usually used to deal with special phraseology.

Step (1) and step (4) are useful, for example, to analyse quantity expression. In Japanese there are some variants of quantity expression with quite the same meaning; for example, the following all of which express "a stone that weighs 100 grams".

"100 gram no ishi"
"omosa 100 gram no ishi"
"100 gram no omosa no ishi"

In the preprocessing section, J-Analyser generates special symbol **Q** just after number strings, and the system treats **Q** as a content word. $FRULE and $MRULE of **Q** look upon "omosa", "100" and "gram" as specifiers for **Q** and reduce these expressions to the same structure as follows.



$FRULE,B and $MRULE,B can't treat a situation of this sort where a dependent appears after its governor.

Step (2) can deal with, for example, such a fact that "an adverb can be modified only by adverbs" which is described as $FRULE of ADVERB.

Step (3) is useful, for example, to analyse such a usage as "ike*ba* iku*hodo*" (*the* farth*er* one goes, *the more*···). This usage can be easily described as $FRULE of the function word "hodo".

Step (6) is useful, for example, to bind words which are hyphnated together. The rule that words before and after "–", where "–" is regarded as a function word, should be bound together can be described as $MRULE of "–".

There are declarative type rules which are called up by these procedural type rules especially by $MRULE linked to phrase category. They are the most significant and the most frequently used representations for binding structures. They are divided into two types. One is an acceptor type rule ($FRULE,B) and the other is a donor type rule ($MRULE,B). They are both linked to a content word. They are basically a declarative type, but they are allowed to describe procedurally their needs.

$FRULE,B is just a generalized case frame of a content word, and it is composed of phrase variables. A phrase variable comes out as some S_i in a phrase representation (see [2]), and is expected to be replaced by a phrase which meets the binding condition that the phrase variable expresses. It takes the following form.

$(p_1 p_2 A \ G)$
or $(p_1 p_2 A \ (n_1 \cdot G_1)(n_2 \cdot G_2)\cdots)$
where [In the following, $FRULE,B of *cw–f* is in mind. So this phrase variable is supposed to be one of S_i–*f*.]

p_1...Penalty point for ellipsis. If there is not an expression in the surface that corresponds to the phrase variable (i.e. if it is abbreviated), the goodness of the phrase *fp* shall be lowered by p_1. If a referent for the ellipsis is identified at the context analysis section (see [4–5]), this penalty point shall be recovered in some ratio.

p_2...Penalty point for embedding. When *cw–f* is a verb, some S_i–*f* can be specified by the verb phrase itself embedded in a sentence ("rentai" structure). But some S_i–*f* is hard to or can't be specified in this way, and p_2 represents this hardness. If p_2 is NIL, such an embedding construction is prohibited. This helps to determine which S_i–*f* is specified by an embedding structure.

A...It points to a certain group of function words (postpositions) which play a role of binding hand. Some function word in the dependent phrase (some *fw–m*) must be a member of the group. Hereby the adequacy of the binding hand of the dependent phrase is ascertained. When the phrase variable is replaced by the phrase *mp*, the phrase category *pc–m* shall be replaced by "*A*".

G...A set of semantic features or a certain program. In the former case, the semantic features of the dependent phrase (*mp*) should contain at least one of *G*. If so, the goodness of the binding is 100. (Semantic features of *mp* are those of *cw–m* and those held by *cxt–m*.) In the latter case, the evaluation of the program which states a certain condition on a dependent phrase (*mp*) should bring some positive number value. The goodness of the binding is this value.

G_i...Same as *G*. Provided that; if the condition given by G_i is satisfied, then n_i is the goodness of the binding and the succeeding part is neglected, otherwise $(n_{i+1} \cdot G_{i+1})$ is examined.

For example, $FRULE,B of the word "motozuku" has two phrase variables (Fig. 3). In the first variable; p_1=**C**, p_2=**A**, *A*=**ga**: **d**, *G*=(": 13"": 11"), and in the second one; p_1=**C**, p_2=**Z**, *A*=**ni**: **o**, *G*=(": 13"": 11"). Where **C**=5, **A**=0, **Z**=NIL, **ga**: **d**=(ga, wa, mo,···), **ni**: **o**=(ni), and ": 13", ": 11" are semantic features. [As to semantic features, J-Analyser now basically utilizes classification numbers in "Word List by Semantic Principles" [6] This rule describes the condition on two dependents of the word "motozuku"; :*d* **ga** :*o* **ni** motozuku (:*d* be based on :*o*).

$MRULE,B describes a binding condition from the side of a dependent phrase, while $FRULE,B does it from the side of a governing phrase. It takes the following form which is similar to that of a phrase variable.

$(A\ G)$

or $(A\ (n_1 \cdot G_1)(n_2 \cdot G_2)\cdots)$

where A, G, G_i, n_i are all identical to those of a phrase variable, except that G and G_i does state a condition rather on a governing phrase than on a dependent phrase.

For example, the word "sansei" (Fig. 3) has a usage such as "kahansuu no *sansei de* kimeru" (to decide something *supported by* majority). Here, the predicate "kimeru" does not obligatorily require a dependent such as "sansei de", and this binding relation is naturally described by using $MRULE,B linked to "sansei"; the rule says that "sansei" accompanied by a function word "de" has a potency to bind with a predicate such as "kimeru".

4. Analysis of Sentences

It is one of the important problems for a natural language analysis system to resolve a degeneration of interpretations of a sentence. (Here we use the word "degeneration" instead of the word "ambiguity".) Most of analysis systems deal with this problem by incorporation of a backtracking mechanism into the system. But leaving every non-deterministics to backtracking does not seem to be a natural modeling of the language undestanding process, and it is not at all an efficient way of processing. The backtracking method, if it is inevitable to use, should be applied as locally as possible.

Degenerations of interpretations are caused by;

(1) Existence of homonyms.
(2) Variety of binding categories.
(3) Variety of binding combinations.
(4) Variety of reference and ellipsis identifications.

(1), (2) and (4) can be treated by the use of the breadth first method, but case (3) can't.

In the Japanese sentences case (3) occurs in an embedding structure, or in a structure where several nouns are connected with "no" such as "N1 no N2 no N3 no$\cdots$" [$\cdots$of N3 of N2 of N1], or in some other structures. J-Analyser processes these "twisted structure" with peculiar algorithms. These algorithms are based upon the depth first method, but the deatails are omitted in this paper [1, 2].

```
(motozuku  motozuku  $di$  motozu  $5k
   (":2111")                                  ... semantic feature
   ((CAga:d ":13" ":11")(CZni:o ":13" ":11"))  ... $FRULE,B
   ($TF,B=E . (":d" be based on ":o")) )       ... translation rule
                                                   for $FRULE,B
(sansei  sansei   $n$
   (":13532" $sa )                            ... semantic feature
   ((BAnog:a :human ":127")(BA!ni:g ":13"))    ... $FRULE,B
   (de:sasae ":23133")                        ... $MRULE,B
   ($TF,B=E . (COND((MEMQ 'suru  %AP-S)       ... translation rule
                    '(":a" endorse ":g"))          for $FRULE,B
              (T
               '(endorsement of ":a"))))
   ($TM,B=E . (supported by ":a")) )          ... translation rule
                                                   for $MRULE,B
```

Fig. 3 Examples of word description.

J-Analyser treats case (1), case (2) and case (4) with the breadth first method. At that time the "goodness" of a phrase is utilized to abandon phrases with very low plausibility.

Fig. 4 shows the outline of the analysis process of J-Analyser. It is composed of four sequential sections; the segmentation section, the reorganization section, the transformation section and the dependency analysis section. The dependency analysis section has as its subsection the context analysis section.

J-Analyser sequentially performs operations on words groups on IQ (input que, see Fig. 4), putting the product on OSTK (output stack, see Fig. 4). What OSTK holds at the end is the result of the analysis. If OSTK holds, at the end, a single object and it is composed of a single phrase, then J-Analyser terminates in success with a unique interpretation. But if it is composed of more than one phrase, it terminates with plural interpretations. And if OSTK holds, at the end, more than one object, then J-Analyser terminates in failure. When there results plural interpretations, they are ordered by "goodness" of phrases.

4.1 Segmentation Section

This section recognizes an independent word and its annex words out of strings on IQ. Here the independent word is splitted into its homonyms, while the annex part is supposed to be uniquely analysed by making use of connection rules between annex words.

An input sentence of J-Analyser is written in Romanized Japanese and is supposed to have spaces placed at least between two independent words. So a string on IQ

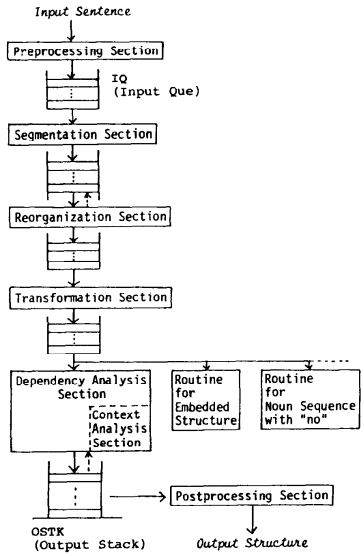


Fig. 4 Flow of the analysis.

is supposed to contain at most one independent word.

Fig. 5 shows that an input string "shouchoudearu" is segmented into "shouchou", "da" ("de" is a conjugation of "da") and "aru". And the independent word "shouchou" is splitted into 3 homonyms.

4.2 Reorganization Section

Although independent words are all content words and most annex words are function words, there are some annex words that should be viewed as content words (Fig. 6). In addition there are some function words that suggest certain kinds of abbreviations.

This section modifies the output of the segmentation section in view of these situations, and reorganizes the contraposition between an independent word and annex words into the contraposition between a content word and function words. Those modification rules are written in the form of computer programs and are linked to the corresponding annex word.

Fig. 7 shows that the annex word "da" is regarded as a content word and the output of the segmentation section is replaced.

4.3 Transformation Section

Some function words signal the analyser to transform case structures on the surface level. There are nearly 20 types of such transformations as passive transformation, causative transformation and so forth. Transformation is, to be concrete, alterations of "*A*" in $FRULE, B or addition of some phrase variables to $FRULE, B.

This section first duplicates $FRULE, B of a content word, and then transforms them according to transformation rules which are linked to such signaling function words. Transformation rules are described declaratively.

An output of this section is a set of phrases whose specification terms are all phrase variables. It is generally a set of phrases, because of the existence of homonyms a content word generally has more than one $FRULE, B.

Fig. 8 shows that $FRULE, B of "shouchou (suru)" is transformed into its passive form because of the existence of a function word "reru". In this case, **ga: o** and **wo: g** are altered to **ny: o** and **ga: g** each;

:*o* ga :*g* wo shouchousuru. (:*o* symbolyzes :*g*)
:*o* niyotte :*g* ga shouchousareru. (:*g* be symbolized by :*o*)

4.4 Dependency Analysis Section

The Japanese sentences are normally governed by the following two principles;

(1) A group of words (a phrase) always specifies what comes afterwards.
(2) Two binding relations never intersect each other.

This section treats a "non-twisted structure". For such a structure we assume the following two principles in addition to the above;

(3) Without considering the succeeding part of an input sentence, it can be determined whether two phrases are in a binding relation or not.
(4) Phrases which have more binding relations are regarded as more plausible interpretations than others.

These principles justified the following procedure of dependency analysis for a "non-twisted structure". This section manages the output stack (OSTK) of outputs for this section from the preceding parts (Fig. 9).

(1) Let *fpl* be the current output of the transformation section (a set of candidate phrases for governor), and *mpl* be the one popped up from OSTK (a set of candidate phrases for dependent).
(2) Pick out a phrase in *fpl* and pick out another phrase in *mpl*, and examine if they satisfy binding

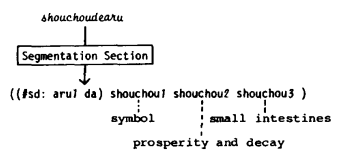


Fig. 5 Segmentation Section.

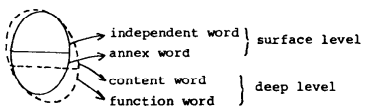


Fig. 6 Classification of words.

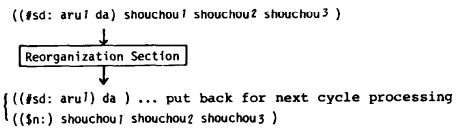


Fig. 7 Reorganization Section.

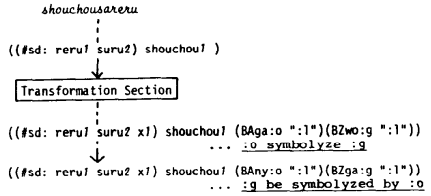


Fig. 8 Transformation Section.

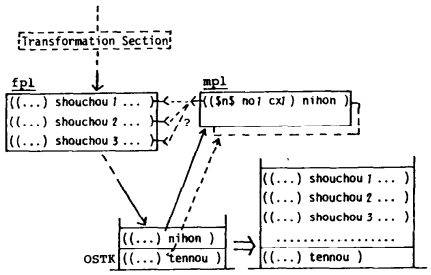


Fig. 9 Dependency Analysis Section.

conditions (see [3]). If a binding relation is established, make subsequently a context analysis (see [4, 5]). Perform this operation on all combinations of phrases in *fpl* and *mpl*.

(3) When there is no combination that establishes a binding relation, stack the current *fpl* onto OSTK and stop. Otherwise, let new *fpl* be the set of the established phrases (but it is restricted to the phrases whose "goodness" are not lower than the highest one by some given allowance), and new *mpl* be the one popped up again from OSTK. And then go to (2).

There is a special register linked to a governing phrase that holds the chronicles of the established bindings. The chronicle register is utilized to analyse binding relations such as juxtaposition that depend on the former established bindings.

4.5 Context Analysis Section

At the time that two phrases are bound together into one phrase, contextual matters are brought forth. This section deals with contextual matters and is placed as a subsection of the dependency analysis section. The products of the context analysis are held on the context register.

There are various topics that should be treated at this section, but for the present, J-Analyser deals with "goodness" of a phrase, tries to solve a reference identification problem, and so forth.

"Goodness" is a numerical representation of plausibility of a phrase. "Goodness" of a simple phrase which has no dependent phrase is supposed to be 100. "Goodness" of a phrase which has dependent phrases is supposed to be $(D+B)/2-P$; where D is an average of "goodness" of those dependent phrases, B is an average of "goodness" of each bindings which are described in each binding rules and P is a sum of penalties for ellipses. Thus there are not only completely acceptable phrases (goodness $=100$) or completely unacceptable phrases (goodness $\leqq 0$) but also fuzzily acceptable phrases ($0<$ goodness <100). An ambiguous expression is analysed into phrases with various "goodness".

For example, \$FRULE,B of verb "furu" (to fall) involves two goodnesses of binding. Because a verb "furu" commonly governs "ame" (rain), "yuki" (snow), "arare" (sleet) and so forth, the goodness of binding in such a common case is supposed to be 100. And in the case that the dependent is other physical object, the goodness of binding (i.e. the commonness of the usage) is supposed to be 60. By this \$FRULE,B, the following expression that is ambiguous because of a word "ame" with two meanings (rain, candy) is splited into two interpretations with different goodnesses.

"Ame ga furu"...It rains. $=100$
A candy falls. $=60$

Similarly, by \$FRULE,B of a verb "nomikomu" (to swallow), the following ambiguous expression is splitted into two interpretations.

Ame wo nomikomu"...To swallow a drop of rain. $=80$
To swallow a candy. $=100$

And by using these goodnesses of phrases, the goodnesses of the following combined expression are calculated as follows.

"Futte kuru ame wo nomikomu"
...To swallow a drop of rain that is coming down. $=90$
To swallow a candy that is coming down. $=80$

J-Analyser cuts down phrases whose "goodness" are lower than the highest one by some allowance. (This allowance is given as a parameter.) Thus J-Analyser aims to avoid prematurely cutting down some allowable interpretations, and at the same time it aims to prevent combinatorial explosion.

Reference identification problem is a big topic in context analysis. There are two types of references; one is an explicit reference using pronoun, and the other is an implicit reference using ellipsis. A phrase variable that is not replaced by any phrases and is left as a variable is regarded as an ellipsis term.

The procedure of a reference identification is in short a variation of dependency analysis. In general, when a dependent phrase is a sentential phrase and a governing phrase involves some references, the identification process shall be invoked. The candidates for referents are the phrases in the sentential phrase and the reference resolvents of the preceding steps which are held on the context register of the sentential phrase. As for a pronoun reference, these candidates are examined if they can establish the same binding relation as the one that the pronoun has established. And as for ellipsis, these candidates are examined if they meet the binding condition which the ellipsis phrase (i.e. the phrase variable) requires.

5. An Application...Structure-to-Structure Translation

As an application, translation into another language is attempted. An input of the translation system is a semantic structure (a phrase) produced by J-Analyser, and the output is another type of a semantic structure expressed in the target language. Fig. 10 is the translation of Fig. 1 into English.

The translation system scans an input phrase applying translation rules which are linked to content words and function words. This is just the process of building up the target structure. A translation rule is just a translation of a binding structure. It corresponds to each binding rule. So, in general, the same word is translated differently if the word is held by a different binding rule. In Fig. 3, the successor of \$TF,B=E is the translation rule into English corresponding to \$FRULE,B and the successor of \$TM,B=E is the one corresponding to \$MRULE,B. In a translation rule a source structure can be reffered through various registers (%AP-S in Fig. 3 is one of them), and a target structure built so far can be reffered

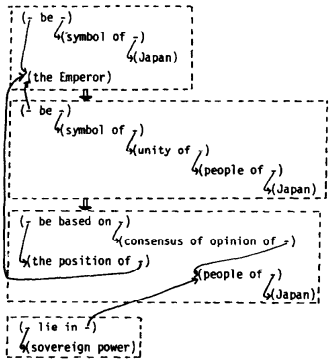


Fig. 10 Translation of Article 1 of the Constitution of Japan into English···(see Fig. 1). *"The Emperor shall be the symbol of the State and of the unity of the people, deriving his position from the will of the people with whom resides sovereign power."*

as well and it also can be modified.

Thus a translation rule is not a simple word-to-word correspondence but is a structure-to-structure correspondence and sometimes a structure manipulating function. In this way it aims to choose the most appropriate correspondences between the languages.

To obtain really natural and plain sentences from the produced semantic structures, it requires more and more carefull choices of words, appropriate pronominalizations and abbreviations. It is basically a creative task and is considered to be very difficult for a computer. It needs further investigation to build such a generation system.

6. Concluding Remarks

J-Analyser is designed for the analysis of a Japanese sentences, and it is not directly applicable to other languages. It intends to faithfully reflect the nature of the Japanese language. It is not grounded upon word order analysis or automaton-like model, it is grounded upon dependency analysis and stack model. It is not based on rewriting rule representation of grammar, it is based on binding structure representation of words. The algorithm of J-Analyser is one-pass on-line.

There are some other approaches grounded upon dependency analysis [3–5]. As to dependency rules, they usually use only case structures. In this paper, not only usual case structure are considered but various kind of binding relations are presented. There are not only acceptor type rules but also donor type rules, and they can be connected with not only content words but also function words and phrase categories. Various phraseology can be naturally described by these rules. Predetermined order of applications of these rules makes it possible to treat easily an exceptional use.

J-Analyser introduces "goodness" of a phrase. On dealing with ambiguous expressions, it aims to avoid prematurely cutting down some allowable interpretations and at the same time it aims to prevent combinatorial explosion by making use of "goodness".

J-Analyser is in its first stage, and especially for the concrete description of each individual linguistic rule, it needs more advanced and precise investigation into the

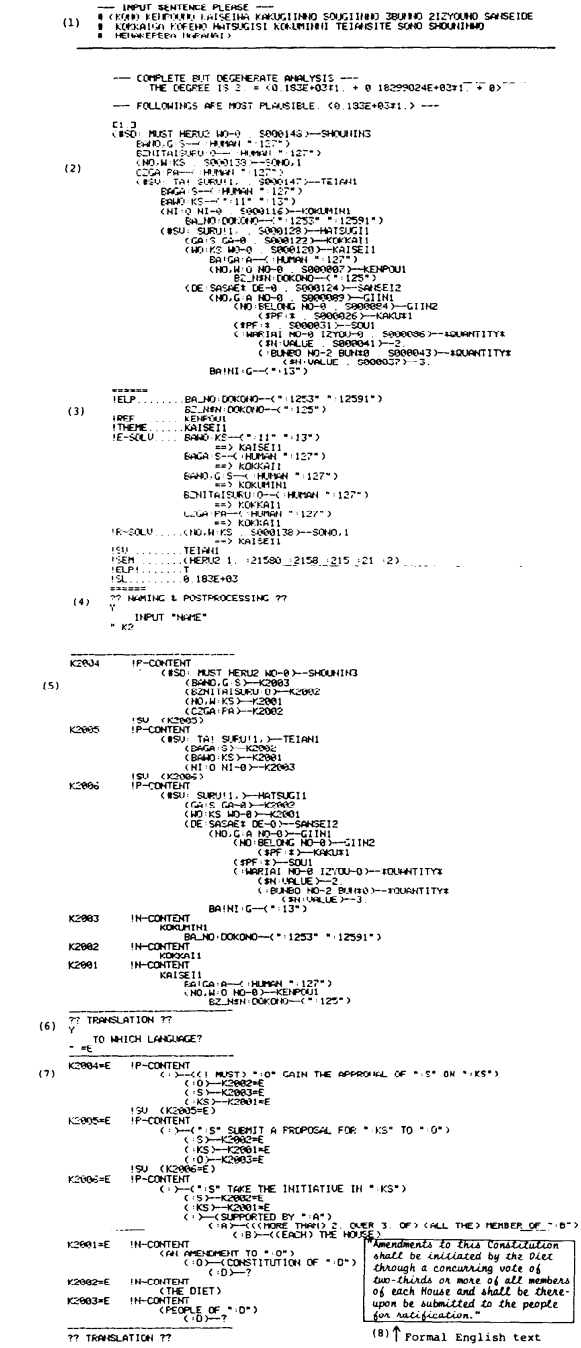


Fig. 11 An operation example of J-Analyser for Article 96 of the Constitution of Japan.

language phenomena. The problems are the systematization of the semantic features and of the phrase categories (or the typology of sentences), the more precise knowledge about the workings of the function words and about the mechanism of references, and so forth.

J-Analyser is now implemented on TOSBAC-5600 computer of ETL with Lisp 2.0. It takes a few seconds to analyse one sentence (without garbage collection time and with not-compiled program.) Fig. 11 shows a raw output of the system. In Fig. 11, (1) is an input sentence from the terminal, (2) is the result of the analysis (the most plausible phrase is presented), (3) is the content of the context register S000148 in (2), (4) is a direction to postprocess the result to get more pretty presentation, (5) is the result of the postprocessing, (6) is a direction to translate the result into English and (7) is the result of the translation. (8) is a formal English text corresponding to (1).

Acknowledgement

The author wishes to express his thanks to Dr. K. Sato who is the head of the Control Div. ETL, the colleagues of that Division and also Dr. H. Inoue of Tokyo Univ. for their encouragement throughout the research. He also wishes to express his thanks to those who have developed and maintain EPICS and its Lisp system of ETL. And he thanks the referees of the "Journal of Information Processing of Japan" who gave the author valuable advice on presentation.

References

1. IKEDA, T. The representation of Japanese words and the Analysis of Japanese Sentences, TG AL-78-32, IECE (1978).
2. IKEDA, T. On Semantic and Syntactic Analysis of Japanese Sentences, *Bulletin of ETL*, (1978) 42–6.
3. NAGAO, M. et al. Analysis of Japanese Sentences by Using Semantic and Contextual Information, *J. IPSJ*, (1976) 17–1.
4. AMEMIYA, M. et al. Japanese Question-Answering System on the Topic of Figure World, *J. IPSJ*, (1977) 18–8.
5. YOSHIDA, S. Syntax Analysis of Japanese Sentence Based on Kakariuke Relation between two Bunsetsu, *Trans. IECE*, (1972) 55D-4.
6. National Language Research Institute Publications—Source 6, 1964, Word List by Semantic Principles, Syuei Syuppan.

(Received July 25, 1979: revised August 6, 1980)