ICOT Technical Report: TR-494

TR-494

Natural Language Processing in the Experimental Discourse Understanding System DUALS-III

by
R. Sugimura (Matsushita), K. Akasaka,
Y. Tanaka, K. Hashida, K. Mukai
& Y. Kubo

July, 1989

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

Natural Language Processing in the Experimental Discourse Understanding System DUALS-III

Ryoichi Sugimura
Matsushita Electric Industrial Co.,Ltd

Kouji Akasaka, Yuichi Tanaka, Koichi Hasida,
Kuniaki Mukai, and Yukihiro Kubo
Institute for New Generation Computer Technology (ICOT)

1 Introduction

In our research on natural language processing (NLP), we have focused on discourse understanding, because our goal is to realize a verbal interface between humans and computers.

The research in the three-year initial stage attempted to recast in terms of logic paradigm the current methods for NLP, to work out the possibilities of further progress, and problems contingent on such an approach.

Along these lines, we started the development of an experimental discourse understanding system called DUALS-I[1], using Prolog running on a DEC2060 computer, which was demonstrated at the FGCS'84 conference, held in November 1984. DUALS-I was able to read 18 and a vocabulary of 200 words, and answer prepared questions. However, its parser, grammar, dictionary, and problem solver were all quite immature and had almost no expansibility.

Nevertheless, through the development of DUALS, we understood the advantages of implementing an NLP system in terms of a logic paradigm, in that it provides good prospects for structuring the system.

The initial task of the four-year intermediate stage was a total reconsideration of DUALS-I. The parser, grammar, dictionary, and problem solver were all to be revised to be more general versions. In parallel with this task, some functions of semantic and discourse processing [15] were studied. The results of these two activities were expected to be integrated into DUALS-II.

In the second year of the intermediate stage, we completed DUALS-II on some prototype software tools developed for DUALS:

- CIL[2], extended Prolog with a record-like structure

- SAX[3], performing high-speed parsing.

These two software tools had generality to some extent, so we decided, starting from these prototype softwares, to build a collection of NLP softwares (called Language Tool Box or LTB)[19], which would be useful not only for the development of successive versions of DUALS, but also for other applications such as the natural language interface for expert systems and data base systems. With these tools, we started developing of DUALS-III, with 200 sentences including many variety of Japanese sentence types, and a basic vocabulary of 2000 words, which was expected to answer questions that had not been prepared.

In the development of DUALS-III, with a large number of sentences, an ad hoc software design was obviously impossible. This meant that DUALS-III should be a system with generality to some extent.

Needless to say, there has been no attempt to perform discourse understanding of a large scale subject text. Therefore, we first had to decide the basic policy for software design.

In the design of DUALS-III, we had to take account of the fact that the higher the process level (morphological, syntactic, to discourse), the more complex handling it needs.

From this point of view, the higher-level processors must be more general and have a more expressive description framework.

For this reason, we decided on the description for each type of processing as follows:

Morphological analysis part: regular grammar, with little expressive power, but can be processed efficiently

Syntactic analysis part: context free grammar, with more expressive power than the morphological analysis part

Problem solver and discourse processor: constraint propagation, with higher generality and freedom than the other parts

We developed the processors necessary for these descriptions with some development facilities, such as debugger, which would constitute LTB.

Next, in designing of the knowledge base (mainly about Japanese), we made on the policy that the knowledge which should be handled by a low-level processor should be processed low-level process, whenever possible.

According to this policy, we classified linguistic knowledge. For the morphological knowledge, we constructed knowledge suitable for deeper semantic processing than previous research, depending on morphological information. For syntactic knowledge, based on dependency grammar (for which it is easy to develop a prototype), we acquired syntactic and semantic knowledge. For discourse processing and problem solving, we described knowledge, based on STASS [5], which describes all discourse knowledge in the form of constraints.

2 Configuration of DUALS-III

DUALS-III consists of LTB, which performs lower-level processing, and some modules for high-level processing, such as the problem solver, as shown in Figure 1. Each module of DUALS is controlled and supplied with channels to communicate with other modules, by a process called a shell [21]. These channels are represented by a thin arrow in Figure 1. DUALS reads a sentence, 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.

3 Linguistic Knowledge Base

3.1 Overview of The Language Knowledge Base

The dictionary [22] 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 linguistic knowledge base together with our Japanese grammar.

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

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

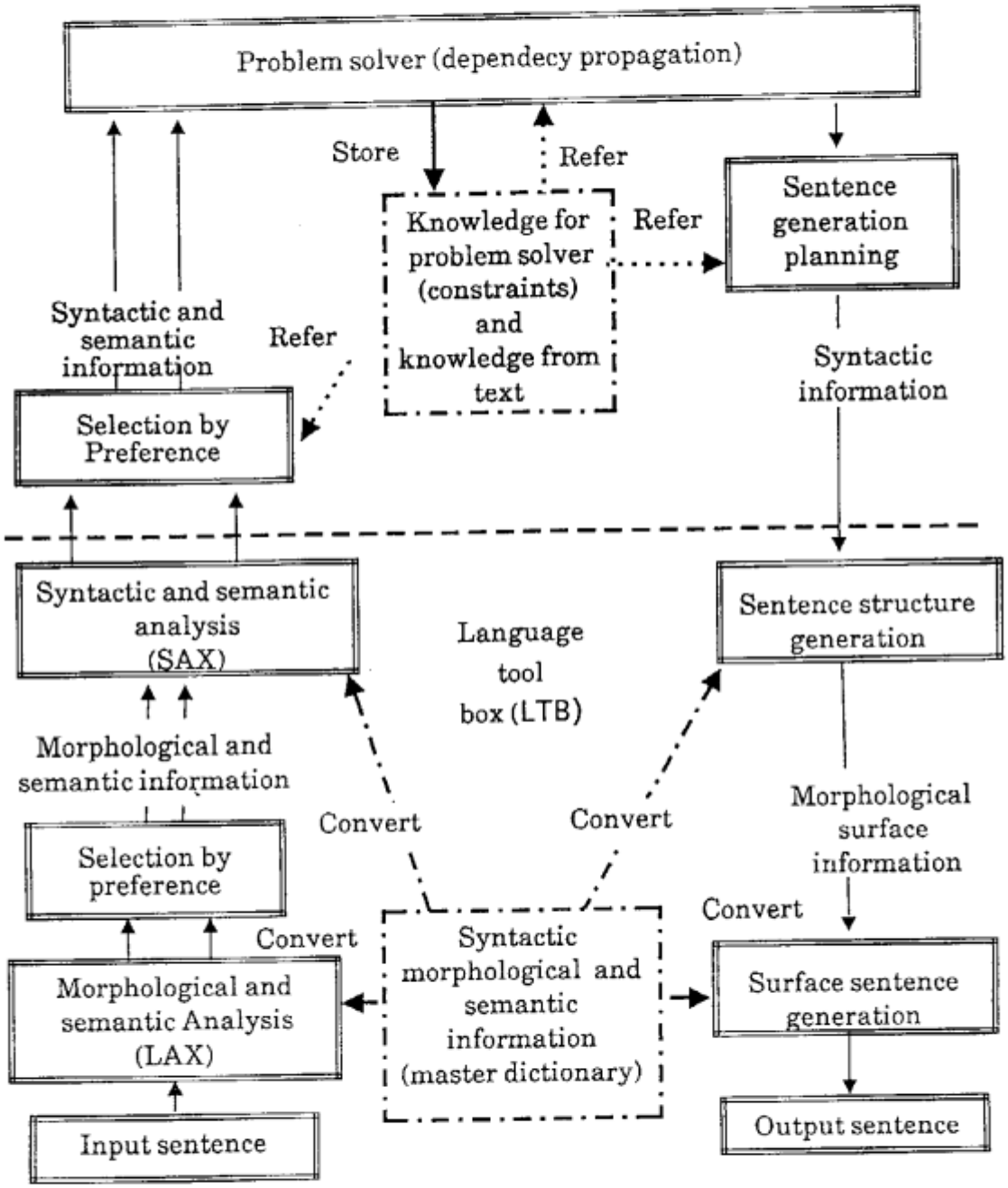

Figure 1: Configuration of DUALS-III

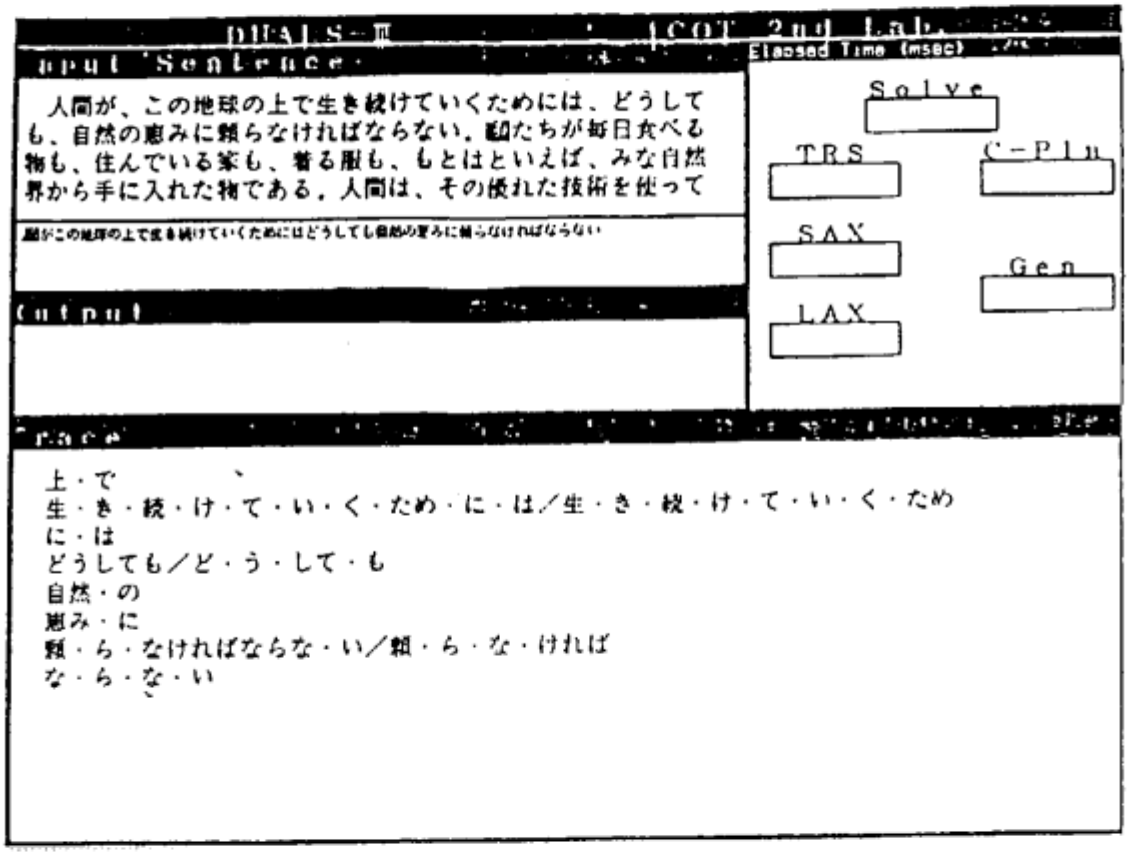

Figure 2: Structure of DUALS-III window

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 standardised notation and consistent knowledge of the system.

3.2 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.

3.3 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 softwares, to implement contextual processing step by step.

However, for the 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 informations for both, there are some discrepancies between the analysis and the generation parts because of the increasing power of discourse processing of DUALS.

We need to improve the master dictionary so that it will reflect the syntactic and semantic information independently acquired in analysis and generation modules.

Secondly, as for the description of knowledge, we have not yet reached the satisfactory one in quantity yet. In the future experiment of DUALS, we need

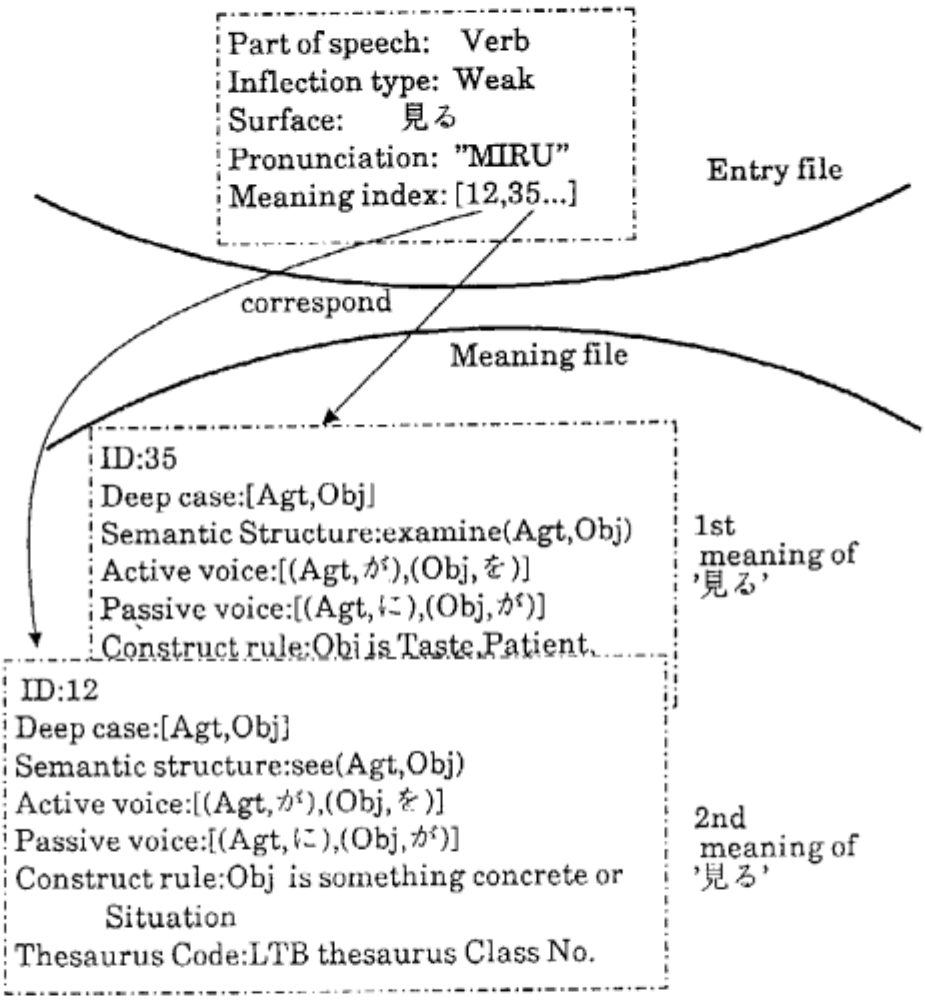

Figure 3: Entry record and meaning record

to acquire sufficient knowledge both in quality and quantity. We also want to try to acquire knowledge through *learning* to enrich the knowledge. Maintaining reservation of consistency of the knowledge base will be the crucial problem.

4 Semantic Description Language

4.1 Overview of CIL

In DUALS, CIL performs:

- Description of Linguistic information [2] [13];
- Semantic processing;
- System description. .

In describing of linguistic information, the Partially specified term (called PST), 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.

4.2 Partially Specified Term and Its Operation

A PST is a record-like structure able to represent a net-work 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 syntactic sugar is given for the PST-operating predicate, which is 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
```

4.3 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 [14]. In other words, it has been theoretically proven that CIL is an extension of Prolog.

At present, we are investigating a type theory and type system, which will enable us to define PST representation and operation in a unified way, for an integrated framework for the three processes performed by CIL. Some of these type systems are currently provided as tools in CIL. We intend to extend CIL aiming at a constraint language, in the future.

5 Morphological and Semantic Analysis

5.1 Overview of the Morphological Analyzing System (LAX)

The morphological analysing program translates a Japanese sentence, which consists of kanji (Chinese characters) and kana (Japanese phonetic characters) and is 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 analyser, which is based on the layered stream technique, which is suitable for both sequential and parallel processing, is developed by the LAX system [18]. The LAX system provides the debugging environment for a morphological analyser and the converter of the dictionary converter.

5.2 Description Format of the morphological and semantic knowledge

As shown in Figure 4, 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 morpheme is exactly in the right-hand feature of the preceding morpheme.

In the previous example, two morphemes 'ka' (word stem of the verb write) and 'ku',(inflectional affix) can be concatenated resulting a verb. The end of the word is recognized by a special feature 'end' , namely, the fact that a morpheme is connected by the feature 'end' with the following morpheme indicates that the morpheme is the last one of the word to which it belongs. After connecting all the morphemes as shown in Figure 4, the LAX analyser constructs the meaning corresponding to each word in the input sentence.

5.3 Special Features of Morphological and Semantic Knowledge

The morphological grammar [4] as the morphological and semantic knowledge, which can be regarded as a regular grammar, is based on Morioka's grammar [6]. Figure 5 shows a sample result of morphological and semantic analysis. In this example, the configuration and the meaning for several derivational words such as adjective 'taka-i',(high) , noun 'taka-sa' (height), transitive verb 'taka-me-ru' (raise), and intransitive verb 'taka-ma-ru' (rise or increase) are constructed from the word stem 'taka', which has a common surface string and meaning to these words.

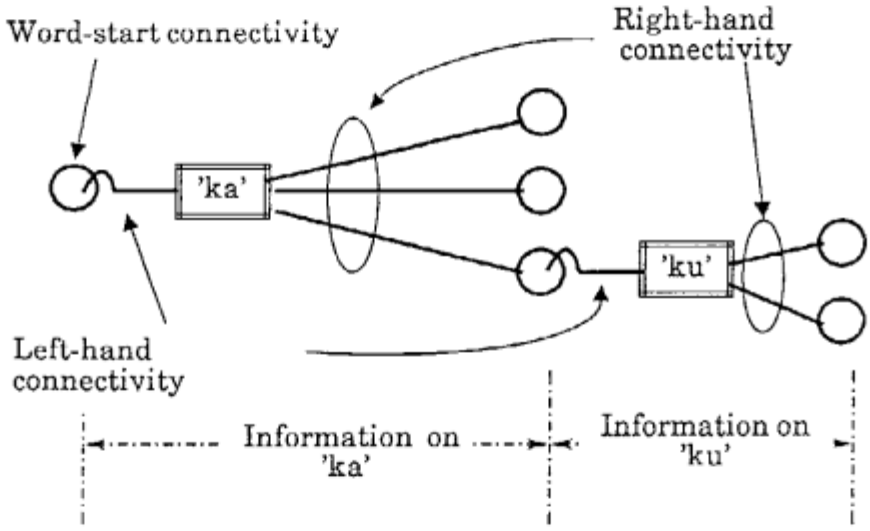

Figure 4. Mechanism of morphological analysis

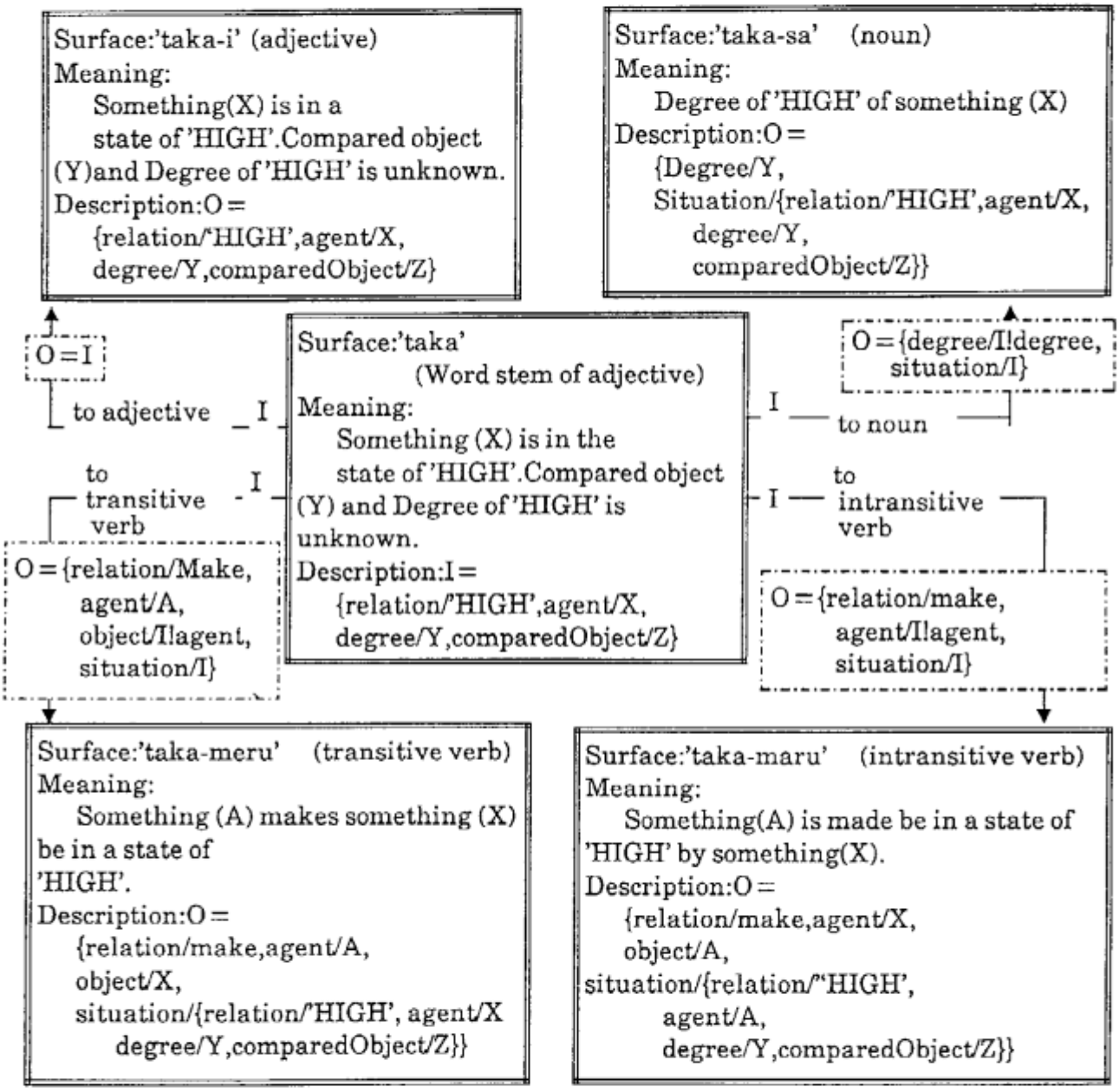

Figure 5: Sample of word stem based morphological analysis

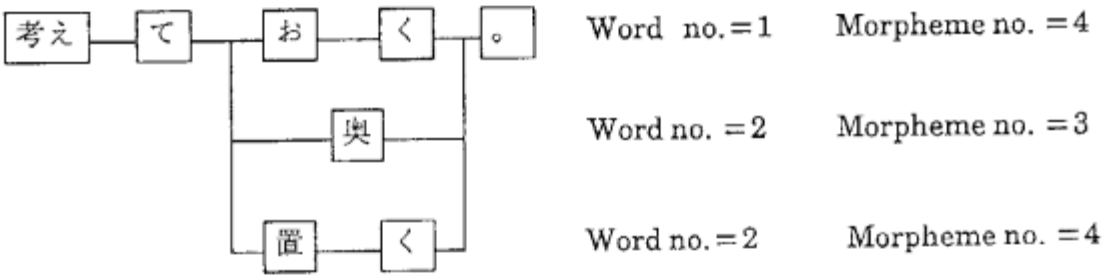

Figure 6: Selection of the best solution

In this method, where the meaning of the word stem 'taka' is written, the semantics for each derivational word can be constructed by applying the derivational rules surrounded by dotted lines shown in Figure 5. 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 shown in the traditional methods, in which each derivational word has its own meaning.

Using this method, the morphological analyser analyses an input sentence and may output several interpretations that are to be reduced to the best one by applying heuristics called the "Minimum phrase method" to those interpretations. When ambiguities exist in spite of the heuristics, the structure corresponding to ambiguities such as those shown in Figure 6 is sent to the syntactic and semantic analyser SAX without data conversion.

Traditionally, a conversion mechanism is needed to propagate the ambiguous result of the morphological analysis to the syntactic analyser. However, in DUALS-III, the morphological analyser is connected with the syntactic analyser naturally because both of them are based on the same mechanism, called the *layered stream mechanism*.[7]

5.4 Future Research on Morphological Analysis

The current morphological analysis program is practical enough even though the result of analysis is ambiguous. It can analyse most sentences that consist of about 30 characters in less than 0.1 seconds. However, the disambiguation mechanism is not very powerful or flexible, so powerful and flexible mechanism needs to be developed. The grammar rules the morphological knowledge should also be extended from the type 3 grammar to the type 2 grammar so as to make the grammar rules flexible.

Classification of morphemes in terms of grammar rules or the description of morphological dictionary is so good that we are planning to increase the entries of the dictionary up to 60,000 morphemes and refine the grammar.

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

6 Syntactic and Semantic Analysis

6.1 Syntactic Analysis System (SAX)

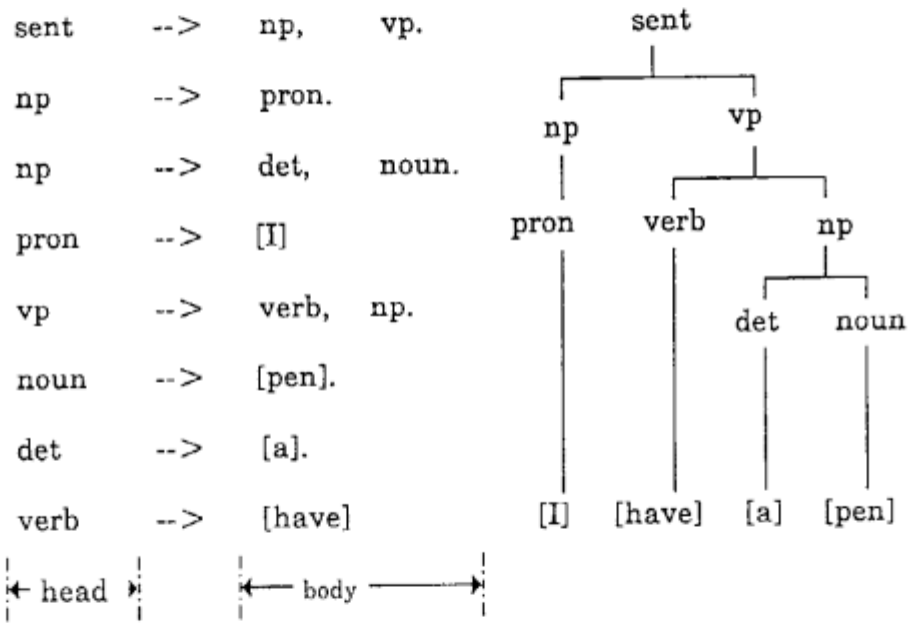

Figure 7: Grammar description using DCG

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.

6.2 Description of Syntactic and Semantic Knowledge

The grammar of SAX is described in extended DCG[8]. DCG is suitable for representing the tree structures as shown in Figure 7. 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.

While the DCG rule has only one symbol in the left hand part (it making type-2 grammar), SAX was extended to type 0 grammar in that more than one symbol can be contained in the head as shown in Figure 8(1), in order to handle the Japanese dependency grammar which includes a type-0 rule, a type-1 rule, and 13 type-2 rules,

This extension is necessary for the description of dependency grammar, which will be explained later.

In many cases, 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 [12] in {pref_rule}. In pref_rule, semantic and syntactic informations can be accessed and from these informations, some preference points in decimal number can be calculated. Moreover, default calculation is executed when no preference rule is specified.

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}[12].

6.3 Special Features of Syntactic and Semantic Analysis

In the development of DUALS-III, we used dependency grammar, which is based on a simple mechanism and matches the grammar writer's intuition, since we needed to build prototype of grammar for subject texts quickly. Dependency grammar produces a graph as the analysed result, so it is not described in type 2 grammar like DCG, which represents only tree structures. For that reason, we extended the SAX grammar description, before developing of the grammar, so that it can describe dependency grammar [20][16].

```
word(Args1),modify_mark,word(Args2)                    (1)
                -->
```

$head_1$,$head_2$, ... ,$head_m$ (1)
--> $body_1$,{$extra_1$} ::{$pref\_rule_1$}, (2)
..... (3)
$body_n$,{$extra_n$} ::{$pref\_rule_n$}, (4)
&{delayed_extra}, (5)
&&{pref_rule}. (6)

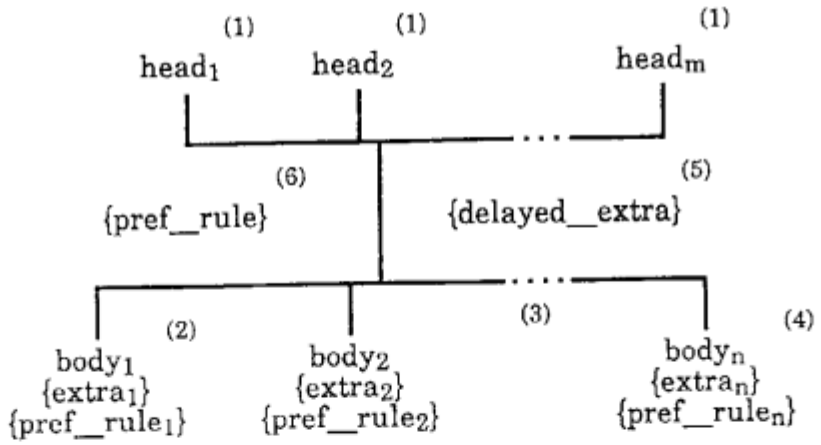

Figure 8: Description of SAX grammar

```
word(Args1),(modify_mark;[]),word_modify_sequence;[]),    (2)
word(Args2),{modify(Args1,Args2)}.                        (3)
```

This rule responds to both adjacent modification and separated modification, as shown in Figure 9. When the word in line (2) can modify the word in line (3), they are rewritten into line (1). Note that the validity of dependency of words is described only in the predicate *modify*, so that the grammar writer must describe only this part as syntactic and semantic knowledge. Because of this feature, syntactic and semantic knowledge can be stored together, making grammar development efficient. Coordination, which is thought to be difficult in dependency grammar, can be handled using a rule based on the tree structure.

6.4 Handling Ambiguity

The syntactic and semantic analysis parts, determines the dependencies of the input word sequence given by the morphological analysis part, using the predicate *modify*. When more than one parse is produced, employing preference-calculation function, the best one is selected. If there is no difference among parses in preference, focusing on the relation of ambiguous meanings (like Figure 11), referring to discourse information, and using constraint logic programming method, the solution is limited [20]. For example, if it is known from discourse information that 'Hanako' (a girl's name) was promised something by somebody, (4) and (5) are selected, which include the information that B= 'Hanako', and they are sent to the problem solver [17][9]. The problem solver which normalizes these constraints and preserves the canonical informations for them, can resolve ambiguity incrementally.

In the existing method, ambiguities have been resolved in parsing with some procedural heuristics. With constraint propagation, the analysis part does not need to limit interpretation unnaturally, as is usual.

6.5 Future Research of Syntactic and Semantic Analysis

The SAX system was extended to allow description of type-0 grammar. However, the development facilities for type-0 grammar are still not satisfactory.

We plan to add useful facilities to our system step by step. For knowledge for analysis, we must make our grammar more general, looking at a variety of subject texts, examining whether the text analysis produces necessary semantic structure for the problem solver or discourse understanding.

The parallel version of SAX (called PAX) is running on the Multi-PSI, parallel inference machine. We plan to shift our system to a parallel machine step by step.

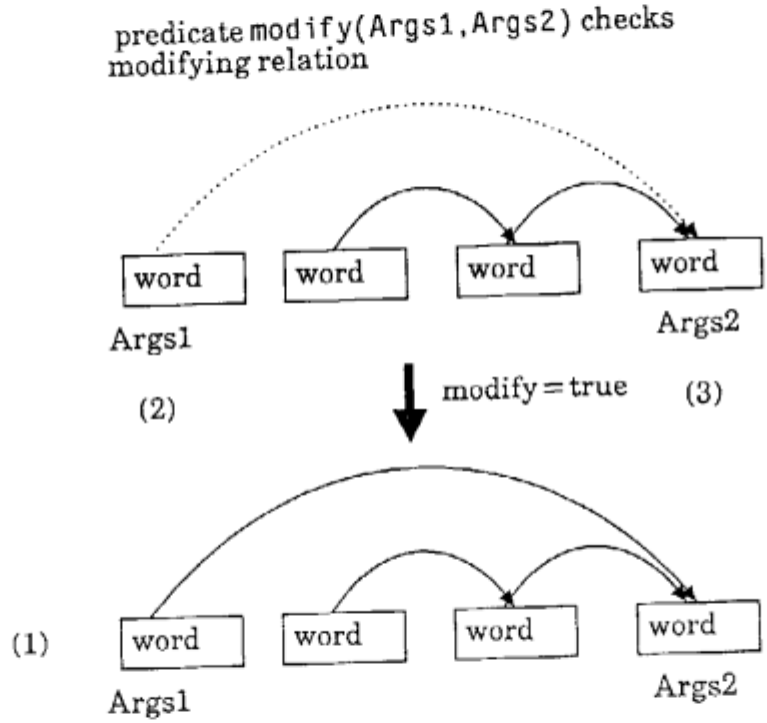

Figure 9: Checking of the modifying relation

'Tarou': boy's name	'Hanako': girl's name	'houkoku -suru': report	'yakusoku': promise	'suru': do or make
'ga': case marker	'ni': case marker		'wo': case marker	

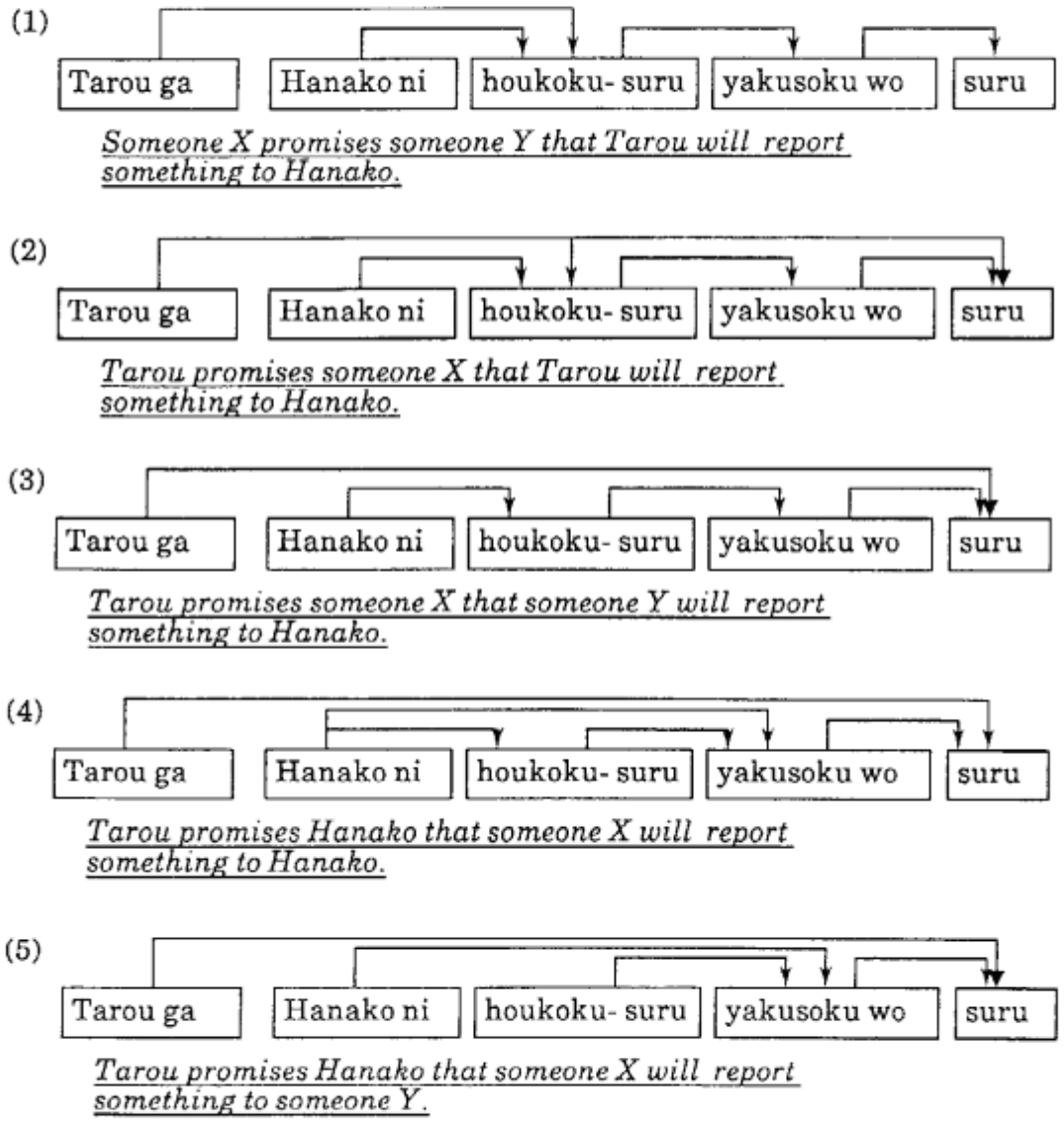

Figure 10: Multiple modification

7 Problem Solving

7.1 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 [11],devised in research on JPSG [9]. 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 [10].

7.2 Description of Knowledge for Problem Solver

This section explains the problem solver, using the honorific relation as example. (Please refer to [11] for technical details.) Assume that the following sentence was uttered in a certain situation [15].

Situation1:Speaker= 'Tarou', Listener= 'Hanako'
(a boy's name) (a girl's name)
Utterance:Jirou-sama ga korareta to Tomoko ga itteimashita
(a boy's name + *respect-marker*)
Tomoko said Jirou 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. 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 interpretation 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,

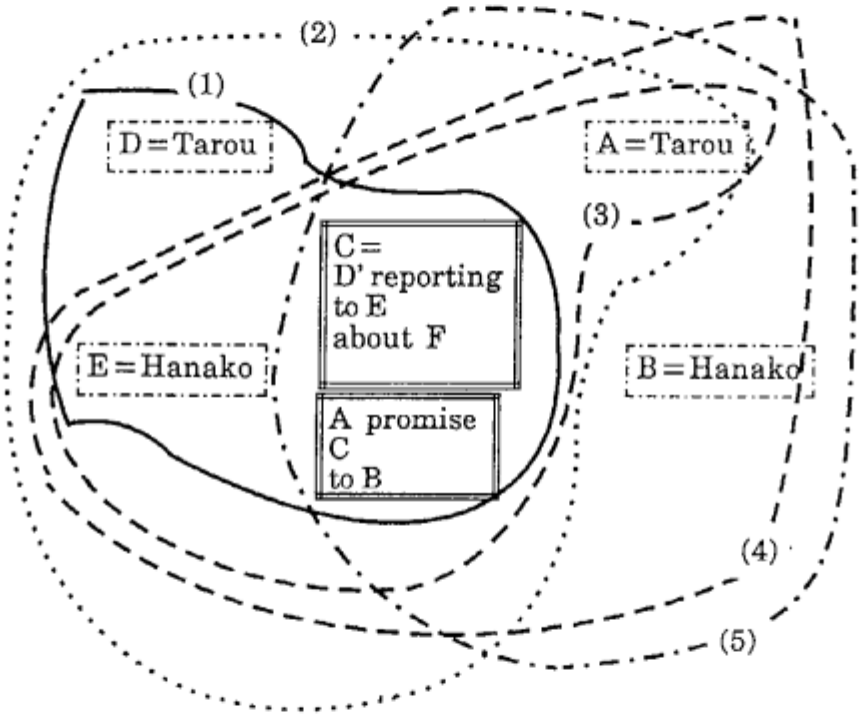

Figure 11: Structure of ambignity

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= 'Tarou', Listener= 'Hanako'
Utterance:Jirou no yatsu kita no desune
Jirou has come, hasn't he ?

From this sentence and the previous one, we know both that 'Jirou' is "Tarou's inferior and that 'Jirou' is "Tomoko's superior.

As shown in the example, the discourse situations with partial informations 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 be not. (as shown in Figure 11). 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 is now essential for DUALS-III, which aims at discourse understanding. For example, if a system happends 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 conclude 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 provide such a function.

7.3 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. In other words, it determines discourse structure, storing the meaning of each sentence as a constraint on the discourse structure [17][9].

This discourse structure involves the utterance itself [15] 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

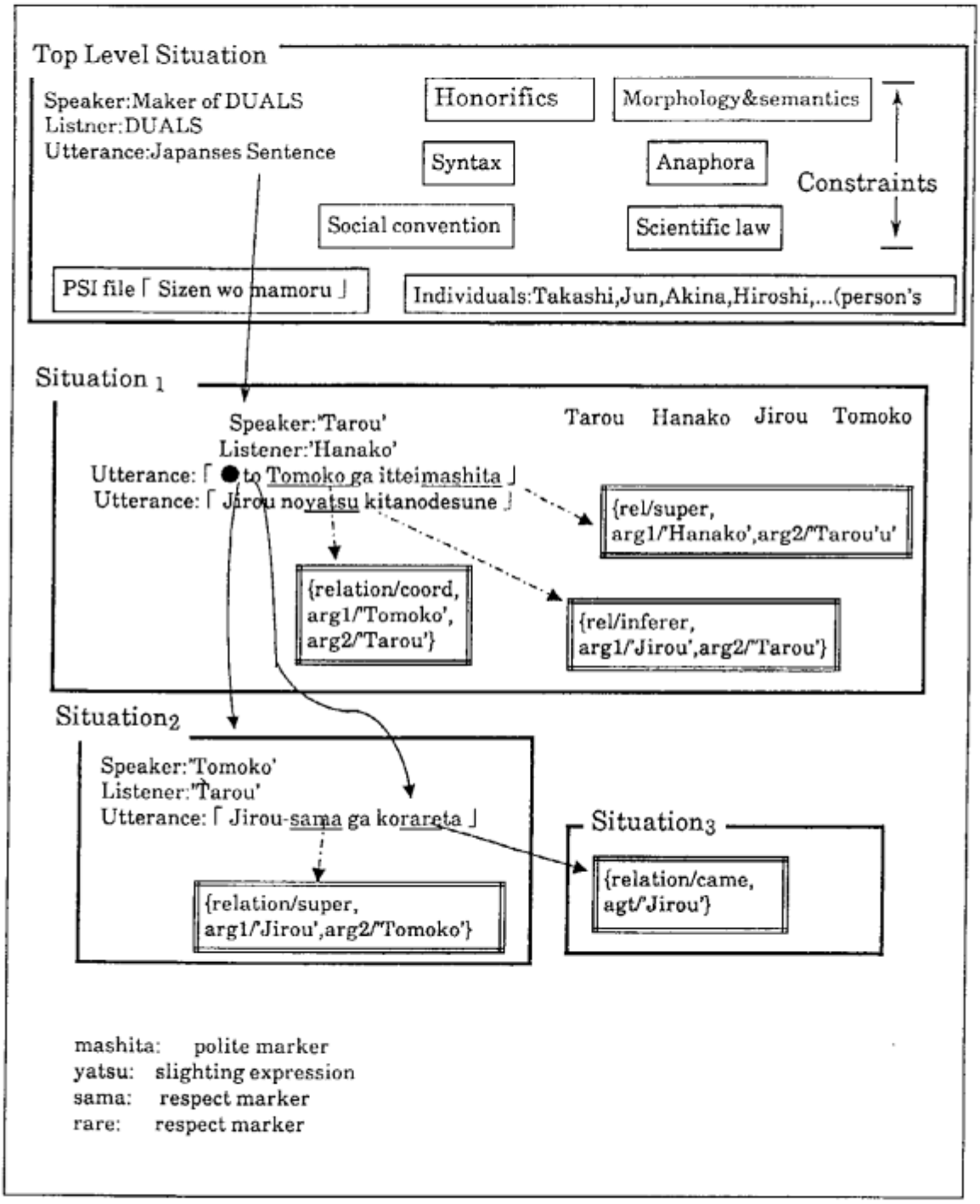

Figure 12: Constraint-based decision of honorific relation

as Situation1), which is a collection of events. Question produces the constraint that 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, 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 13 is produced.

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.

7.4 Future Research on Problem Solving

As already stated, constraint propagation program makes it possible to determines the structure of a discourse, storing constraints in it incrementally. Currently, these constraints are limited to Horn clauses as Prolog. We plan to extend this to first order logic clauses, which involves Horn clauses. In addition, the procedure for reading rules will also be extended so that rules can be read in not only top down but also other ways, to make the system more flexible. At present, we are considering using two approaches at the same time to acquire constraints on discourse structure. One is to store knowledge incrementally, using the current framework of DUALS. The other is to convert once all the current knowledge for both analysis and generation into the problem solver format, and to realize the total discourse understanding mechanism in terms of a constraint logic programming paradigm. Since the latter approach, which provides a good prospect and powerful framework for knowledge description, is suitable for acquiring different types of knowledge related to discourse, we will take this approach, then verify the quality of the knowledge through experiments. Next, the acquired knowledge will be transferred to DUALS using the former approach. In this way, total evaluation, including the processing speed, will be performed. Finally, the knowledge will be transferred to the general Japanese processors, LTB.

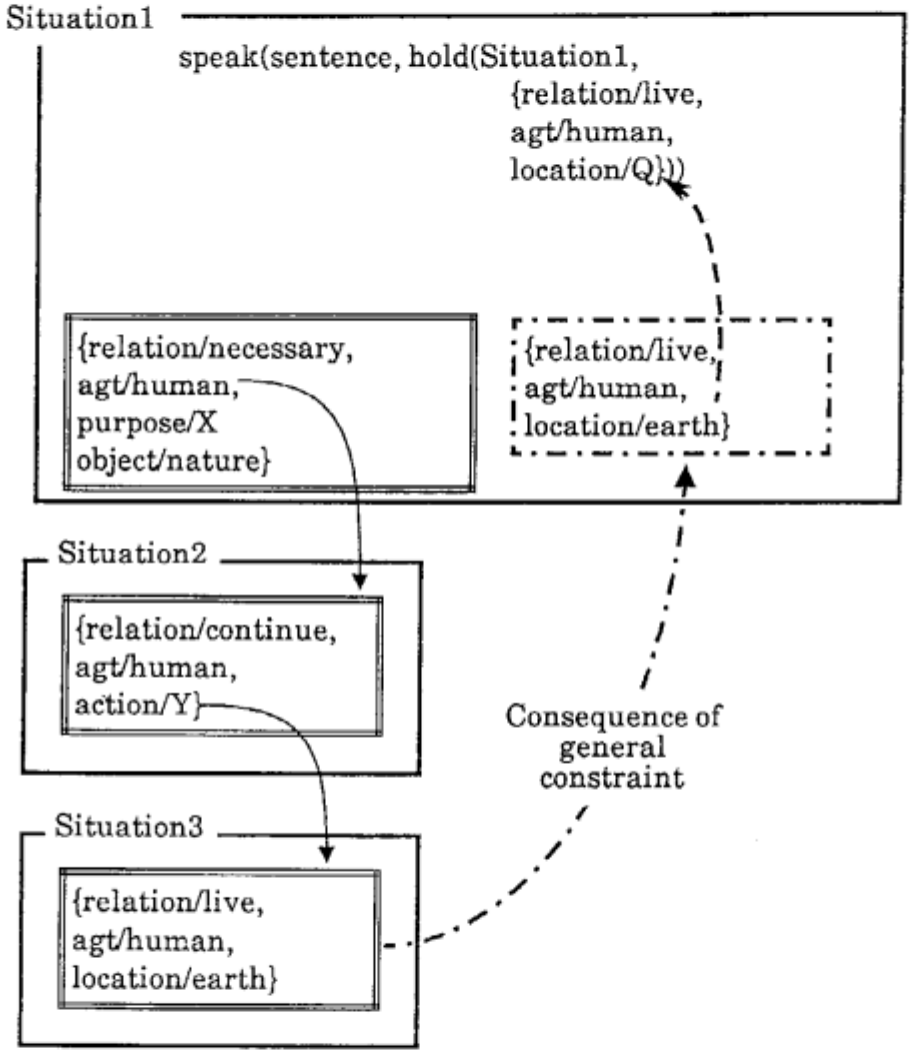

Figure 13: Problem solving sample

8 Conclusion

DUALS-III was demonstrated at the FGCS'88 conference, held in November 1988, as one of the products of the intermediate stage. The research into NLP in ICOT is likely revolve around DUALS-III in the final stage.

Through the development of DUALS-III, it was confirmed that constraint programming and situation theory efficiently elucidate complicated discourse understanding and acquire the required knowledge. Therefore, we need to acquire more constraints for discourse understanding in parallel with establishing a dependency propagation technique. After that, constraints will be transferred to LTB so that it will be more powerful and useful.

The shortage of computational power is likely to happen in the future. So we intend to transplant our system to the parallel inference machine (PIM) to increase the total processing power for natural language.

In the final stage, we will try to realize a DUALS for flexible conversation.

Acknowledgements

We would like to thank Professor Hozumi Tanaka at Tokyo Institute of Technology and the members of the NLU/SG-2 working group for their intensive discussions and suggestions. We would also like to thank Dr. Uchida, the chief, and all the Second Research Laboratory for their valuable discussions. We wish to express our thanks to Dr. Fuchi, the director of ICOT, for his encouragement and support in our work. We would like to thank Dr. Patrick Saint Dizier for his kind advice and many kinds of support to publish this work.

References

[1] Mukai,K, and Yasukawa,H A System of Logic Programming for Linguistic Analysis. In *Cambrigdge,MA: MIT Press*, 1987.

[2] T. Amanuma, T. Suzuki, T. Okunishi, and K. Mukai CIL programming kankyô (CIL programming environment *in Japanese*). In *Proceedings of the 2nd Annual Conference of Japanese Society for Artificial Intelligence*, pages 211–214, 1988.

[3] Y. Matsumoto, and R. Sugimura A Parsing System Based on Logic Programming In *Proceedings of the International Joint Conference of Artificial Intelligence*, 1987.

[4] H. Sano, K. Akasaka, Y. Kubo and R. Sugimura Go-Kousei ni Motoduku Keitaiso Kaiseki (Morphological Analysis with derivation and inflection (*in*

Japanese),). In *Proceedings of the 2nd Annual Conference of Japanese society for Artificial Intelligence*, 1988.

[5] J. Barwise and J. Etchemendy *The Liar:An Essay on Truth and Circular Propositions*. MIT Press, Amsterdam, 1987.

[6] K. Morioka Goi no Keisei (Formation of a vocabulary *in Japanese*) Meiji Shoin, Gendai-go Kenkyuu, 1987.

[7] N. Okumura and Y. Matsumoto Layered Stream wo mochiita Heiretsu-Programming(*in Japanese*),). In *Proceedings of The Logic Programming Conference '87*, 1987

[8] Pereira, F. and Warren, D. "Definite Clause Grammars for Language Analysis – A Survey of the Formalism and a Comparison with Augmented Transition Networks", *Artificial Intelligence 13*, pp.231-278, 1980

[9] K. Hasida Dependency propagation: A unified theory of sentence comprehension and generation. In *Proceedings of the 10th IJCAI*, pages 664–670, 1987.

[10] K. Hasida AI to wa nande naika - on partiality of information (What is not AI) *in Japanese*). *bit*, 20, 1988.

[11] K. Hasida and H. Sirai Zyôkentuki tan ituka (conditioned unification, *in Japanese*). *Computer Software*, 3:28–38, 1986.

[12] Y. Matsumoto and R. Sugimura Koubun kaiseki system SAX no tameno bunpô kijutu gengo (grammar description language for the sax parsing system *in Japanese*). In *5th Conference Proceedings of Japan Society for Software Science and Technology*, pages 77–80, Tokyo, 1988.

[13] K. Mukai Unification over complex indeterminates in Prolog. Technical Report 113, ICOT, 1985.

[14] K. Mukai Partially specified term in logic programming for linguistic analysis. In *FGCS'88*, 1988.

[15] R. Sugimura Japanese honorifics and situation semantics. In *Proceedings of the 11th COLING*, pages 507–510, 1986.

[16] R. Sugimura *Logical Dependency Grammar and its Constraint Analysis*, volume 679 of *ICOT TM*. ICOT, 1987.

[17] R. Sugimura Ronri-gata bunpô ni okeru seiyaku kaiseki (constraint analysis on logic grammars (*in Japanese*)). In *Proceedings of the 2nd Annual Conference of Japanese society for Artificial Intelligence*, pages 427–430. Japanese society for Artificial Intelligence, 1988. a prized paper.