ICOT Technical Report: TR-320

TR-320

Constraint Analysis on Japanese Depending Structure

by
R. Sugimura & K. Mukai

November, 1987

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

Constraint Analysis on Japanese Dependency Structure

Ryoichi Sugimura and Kuniaki Mukai

Institute for New Generation Computer Technology (ICOT)
21F, Mita Kokusai Building,
1-4-28, Mita, Minato-ku, Tokyo 108, Japan

ABSTRACT

This paper describes a working instance of "A System of Logic Programming for Linguistic Analysis" [Mukai 87]. Constraint analysis on Japanese dependency structure based on three-value logic is presented. Facilities for three-value logic in language CIL realizes constraints in dependency structure explicitly and statically in our system DUALS. System for grammar and semantics also presented briefly.

0. Introduction

Unlike European languages, which are based on *phrase structure* presented in context free phrase structure grammars [Kaplan & Bresnan][Pollard], Japanese sentences have a *dependency structure* among their constituents [Nitta 86] [Watanabe 81][Hashimoto 80][Yoshida 72] which is very hard to present in context free rules. *Dependency structure* is ruled by two kinds of constraints which correspond to phrase structure rules in European languages. One is a set of constraints between modifying relations (MRs). The other is a set of constraints to determine whether one phrase can modify other phrases syntactically and semantically. When one phrase modifies another phrase, the semantic structure can be constructed from the modifying and modified phrases. Basically, the construction is carried out by the unification mechanism.

In the field of computer linguistics, it is well known that without contextual information it is very hard to obtain only one meaning of one sentence from many interpretations, but it is also very hard to make a bridge between analysis of one sentence and information of context. There are three kinds of ambiguities. Firstly, there are many ambiguities in dependency structure. Secondly, one word may have more than one meaning. Thirdly, when Japanese is presented only in phonetic characters, there may be lexical ambiguities. The traditional approach to the first kind of ambiguity depends on the heuristics applied in sentence analysis [Tsujii, Nakamura and Nagao 84][Nakamura 86][Ozeki 86]. These heuristics are applied basically to reduce the ambiguity in two-value logic, and there is no room to represent ambiguity explicitly. The main feature of the analysis proposed here is the treatment of MRs in

three-value logic realized by *lazy evaluation programming* in CIL [Mukai 85]. In this framework, ambiguity and lack of information are expressed explicitly as Unbound with constraints.

Disambiguation of usage of word meaning and lexical ambiguity in phonetic characters are not discussed in this paper. Constraint programming enables constraints on ambiguities to be *propagated* to the proper stage in contextual analysis [Grosz 87] and to be handled there.

1. CIL and Three-value Propositional Logic

First, the CIL truth table is as follows.

∧	1	0	U
1	1	0	U
0	0	0	0
U	U	0	U

∨	1	0	U
1	1	1	1
0	1	0	U
U	1	U	U

¬	
1	0
0	1
U	U

~	
1	0
0	1
U	1

For example, the following constraint can be solved in CIL.

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

```
> constr(and(A, B), false).
A = Unbound
B = Unbound
```

Implication A ⇒ B is defined as ¬A∨B.

2. Preparations for the Analysis of Modifying Relations

Generally speaking, Japanese sentences have no delimiter between words, as shown in the following example.

▸ Japanese sentence :

Taro	Hanako	to report		promise
太 郎	が 花 子	に 報 告	す る 事	を 約 束 す る 。

(Taro promises Hanako to report something.)

Suppose that such a Japanese sentence is segmented into unique *PHRASEs* or "bunsetsu". Let us number these *PHRASEs*.

▸ Results of lexical segmentation

	太 郎 が/	花 子 に/	報 告 す る/	事 を/	約 束 す る 。
No.	1	2	3	4	5

Among these *PHRASEs* or "bunsetsu", the dependency structure can be obtained as shown in the following example. In Fig. 2.1, the arrows represent the MRs. For example, the nominative *PHRASE* "Taro ga" modifies the verb "yakusoku suru" (promises). The dative *PHRASE* "Hanako ni" modifies two verbs, "houkoku suru" and "yakusoku suru".

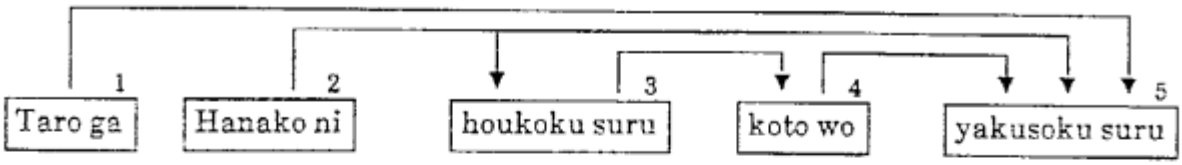


Fig. 2.1

To present modifying relations, the modifying relation table (MRT) is used , in which MRs are represented as matrix elements [Yoshida 72]. The MRT is created as shown in Fig. 2.2 according to the number of *PHRASE*s. The initial value of every element in the MRT is UNBOUND.

1	U_{12}	U_{13}	U_{14}	U_{15}
	2	U_{23}	U_{24}	U_{25}
		3	U_{34}	U_{35}
			4	U_{45}
				5

Fig. 2.2

1	0	0	0	1
	2	1	0	1
		3	1	0
			4	1
				5

Fig. 2.3

Fig. 2.3 is an example of an MRT corresponding to the dependency structure in Fig. 2.1. In Fig. 2.3, numbers 1 to 5 indicate the number of the line and column. 1 indicates a link and 0 indicates that there is no link. For example, 1 in MRT(1,5) (line 1, column 5) indicates that the *PHRASE* in position 1 is connected to the *PHRASE* in position 5.

A special feature of this paper is the realization of linguistic constraints with this modifying relation table (MRT). Details are given in Section 3.

3. Constraints on dependency structure

Our system is built on three-value propositional logic in CIL. In this section, we will present constraints in first-order logic first, and then we will present them in three-value propositional logic.

Two sets of constraints on dependency structures are as follows.

For notation, the first-order predicate M(i,j) means the existence of a modifying relation between *PHRASE* i and *PHRASE* j.

$$M(i,j) = \{T,F\}$$
where $N(i) \wedge N(j) \wedge (i \leqq$ number of *PHRASE*s $- 1) \wedge (j \leqq$ number of *PHRASE*s$)$

In our approach, M(i,j) is realized as logical variable U_{ij} in MRT. In CIL, U_{ij} is realized as follows.

$$Uij = \{1,0,U\}$$
where $N(i) \wedge N(j) \wedge (i \leqq$ number of *PHRASE*s $- 1) \wedge (j \leqq$ number of *PHRASE*s$) \wedge (i < j)$

3.1. Constraints between Modifying Relations

UC1 Regardless of their grammatical nature, all *PHRASEs* except the last should modify at least one *PHRASE* to the right of themselves.

$$(\forall i)(\forall j)((i<j)\Rightarrow(\exists k)((i<k)\wedge M(i,k)))$$

Using CIL for the description of constraints in propositional logic, this constraint is presented as follows. This example describes constraint UC1 in Fig.2.2.

$$(U_{12}\vee U_{13}\vee U_{14}\vee U_{15})\wedge(U_{23}\vee U_{24}\vee U_{25})\wedge(U_{34}\vee U_{35})\wedge U_{45}=1$$

UC2 The last *PHRASE* modifies no other *PHRASEs*.

$$\neg M(n,j)$$
where n = number of *PHRASEs*

It eliminates the logical variable corresponding to M(j,n) from the MRT.

UC3 No two MRs should cross each other.

$$(\forall k)(\forall i)(\forall j)(\,((i<j<k)\wedge M(i,k))\Rightarrow(\forall l)((l<i)\Rightarrow\neg M(l,j))\wedge(\forall m)((k<m)\Rightarrow\neg M(j,m))\,)$$

In CIL, the constraints for MRT in Fig.2.2 are described as follows.

$$(U_{13}\wedge(U_{24}\vee U_{25}))\vee(U_{14}\wedge(U_{25}\vee U_{35}))\vee(U_{24}\wedge U_{13}\vee U_{35}))\vee$$
$$(U_{25}\wedge(U_{13}\vee U_{14}))\vee(U_{35}\wedge(U_{14}\vee U_{24}))=0$$

Therefore, the MRs in Fig. 2.4 are illegal.

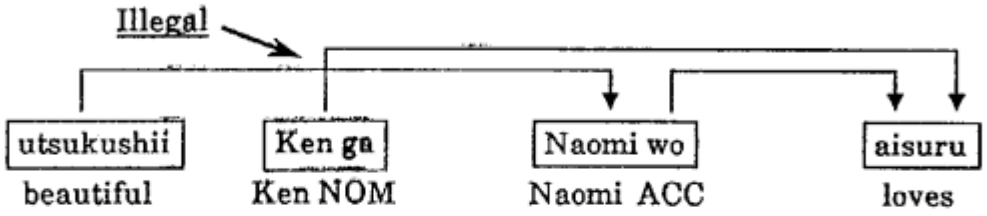


Fig. 2.4

3.2. Constraints between the Modifying *PHRASE* and the Modified *PHRASE*

To determine constraints between the modifying and modified *PHRASE*, the following predicates are defined.

1) Syn(i,j) is the predicate to check whether M(i,j) is true or false syntactically. The relation between M(i,j) and Syn(i,j) is as follows.

$$M(i,j)\Rightarrow Syn(i,j)$$

In our system, Syn(i,j) is realized as a set of predicates like those in the following example.

```
syn(i,j,I) = {0,1,U}
syn(i,j,I) :- syn__ok(syntax(I,i),syntax(I,j))
```

where syn__ok is a predicate to check whether a pair of syntactic features can be connected or not. syntax(I,j) is a function from a set of information of *PHRASEs* to a syntactic feature of *PHRASE* i. In CIL, this information, I, and function syntax(I,i) are realized by a partially specified term (PST) [Mukai 86], as shown in the following example.

```
syntax(I,i) = I ! i ! syntactic-feature = {material/noun,...}
        where
I =     {i/ {phrase-number/1,
             syntactic-feature/{ material/noun
                                 inflection/nominal
                                 surface-case/agent}}
```

2) Sem(i,j) is a predicate to check whether M(i,j) is true or false semantically. The relation to M(i,j) and Syn(i,j) is as follows.

$$\text{UC4)}\quad M(i,j) \Rightarrow Sem(i,j) \Rightarrow Syn(i,j)$$

In our system on CIL, these relations are realized as follows.

$$\neg Uij \vee (\neg sem(i,j,I) \vee syn(i,j,I))$$
$$\text{where } sem(i,j,I) = \{0,1,U\}$$

3) SemE(i,j) is a predicate to check whether semantic information composed from *PHRASE* i and *PHRASE* j exists as partial information of a sentence.

$$\text{UC5)}\ M(i,j) = SemE(i,j) \Rightarrow member(constr\text{-}sem(i,j),Sentence\text{-}Meaning)$$

constr-sem(Ii,Ij) is a function from Ii,Ij to constructed partial meaning. This function is defined in the system to construct semantic information by users. Sentence-Meaning is a set of partial meaning obtained by a function constr-sem.

SemE(i,j) should be distinguished from Sem(i,j). Sem(i,j) only checks possibility. In our system, this predicate corresponds to the construction of semantic information from *PHRASE* i and *PHRASE* j.

SemE(i,j) is realized as follows. Since the truth value of SemE(ij) is equal to Uij, Uij is used to describe constraints. A system to construct semantic information is represented in Section 5.

$$\neg Uij \vee (constr\text{-}sem(Ii,Ij,Iconstr) \wedge member(Iconstr,Sentence\text{-}Meaning))$$

3.3. Realization of Constraint on Dependency Structure

An analysis of the dependency structure can be made. This section analyzes constraint analysis on the example sentence in Section 2.

First, let us put together all constraints. Since this example does not use semantic constraints UC4') will be used for UC4).

UC1) $(U_{12} \vee U_{13} \vee U_{14} \vee U_{15}) \wedge (U_{23} \vee U_{24} \vee U_{25}) \wedge (U_{34} \vee U_{35}) \wedge U_{45} = 1$

UC3) $(U_{13} \wedge (U_{24} \vee U_{25})) \vee (U_{14} \wedge (U_{25} \vee U_{35})) \vee (U_{24} \wedge U_{13} \vee U_{35})) \vee$
$(U_{25} \wedge (U_{13} \vee U_{14})) \vee (U_{35} \wedge (U_{14} \vee U_{24})) = 0$

UC4') $(\neg U_{12} \vee syn(1,2,I)) \wedge (\neg U_{13} \vee syn(1,3,I)) \wedge (\neg U_{14} \vee syn(1,4,I)) \wedge$
$(\neg U_{15} \vee syn(1,5,I)) \wedge (\neg U_{23} \vee syn(2,3,I)) \wedge (\neg U_{24} \vee syn(2,4,I)) \wedge$
$(\neg U_{25} \vee syn(2,5,I)) \wedge (\neg U_{34} \vee syn(3,4,I)) \wedge (\neg U_{35} \vee syn(3,5,I)) \wedge$
$(\neg U_{45} \vee syn(4,5,I)) = 1$

Suppose that $syn(1,2,I) \vee syn(1,4,I) \vee syn(2,4,I) \vee syn(3,5,I) = 0$. --(S).

From UC1),

$U_{45} = 1$. --(A).

From (A) and UC5),

member(constr-sem(4,5), Sentence-Meaning) = 1 --(O1)

From UC4') and (S)

$U_{12} \vee U_{14} \vee U_{24} \vee U_{35} = 0$ --(B)

member(constr-sem(1,2), Sentence-Meaning) = 0 --(O2)

member(constr-sem(1,4), Sentence-Meaning) = 0 --(O3)

member(constr-sem(2,4), Sentence-Meaning) = 0 --(O4)

member(constr-sem(3,5), Sentence-Meaning) = 0 --(O5)

From (B) and UC1)

$U_{34} = 1$

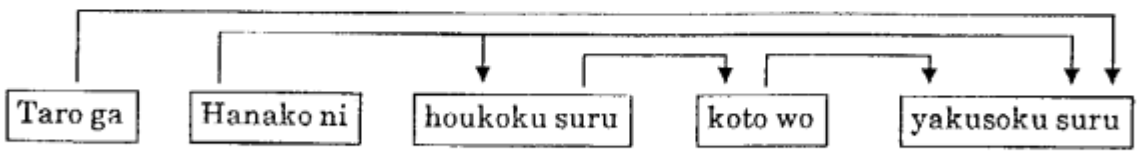
member(constr-sem(3,4), Sentence-Meaning) = 1 --(O5)

Then constraints UC1), UC3), and UC4) are reduced automatically by CIL.

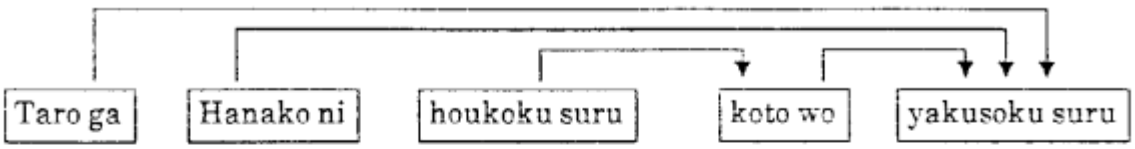
$(U_{13} \vee U_{15}) \wedge (U_{23} \vee U_{25}) \wedge (\neg U_{13} \vee \neg U_{25}) = 1$ --(O6)

These constraints correspond to the following five interpretations.

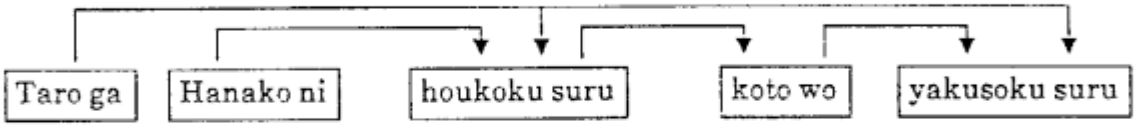
After determining(S), logical variables U_{13}, U_{15}, U_{23}, and U_{25} will be checked by (O6) when they obtain ground value 1 or 0. It is very important to recognize that (O6) is attached to these variables as a constraint realized by lazy evaluation and that CIL does not perform term rewriting. A constraint can be realized by lazy evaluation only if it is used to check whether some logical variable obeys the constraint. Constraints on CIL will not work until some logical variable obtains ground value 1 or 0. CIL will not reduce constraints, but if there found some ground value 1 or 0, CIL will check whether the value obeys the constraint.



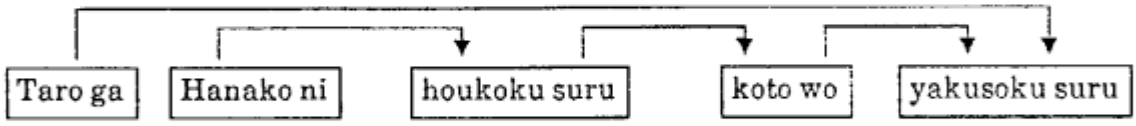
promise(Taro,Hanako,report(agent,Hanako,something))



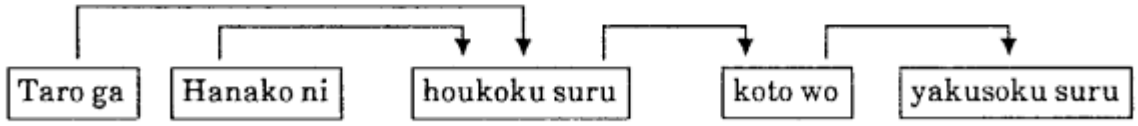
promise(Taro, Hanako, report(agent1, agent2, something))



promise(Taro, agent1, report(Taro, Hanako, something))



promise(Taro, agent1, report(agent2, Hanako, something))



promise(agent1, agent2, report(Taro, Hanako, smething))

3.4 Relationships between SemE and Information in Context

Determining all values for U_{ij} means obtaining unique semantic information of a sentence. Computationally, this means disambiguation of sentence meaning. Now, let us consider the relationships between SemE and information in Context.

First, let us define predicate Sem-Context(I). If information I is in the context, Sem-Context(I) = T, otherwise F.

$$\text{Sem-Context(I)} = \{T,F\}$$
$$\text{Sem-Context(constr-sem(i,j))} \Leftarrow \text{SemE(i,j)}$$

Under this constraint, to determine all values for Uij, *negative information should be set for constr-sem(i,j)* because if ¬Sem-Context(constr-sem(i,j)) then the logical consequence is ¬SemE(i,j).

In our system, we should be very careful about realizing this constraint because a lack of information about constr-sem(i,j) will make Sem-Context false. We think we should distinguish lack of information from explicit negation [Fenstad, Halvorsen, Langholm, and Benthem]. This is main reason why we made our model on three-value logic. So, for this reason, the weak negation ~ is used to define sem-context(constr-sem(i,j),World).

$$\neg \text{Uij} \vee \text{Sem-Context(constr-sem(i,j), World)} = 1$$
$$\text{sem-context(constr-sem(i,j),World)} = \sim\neg\text{member3(constr-sem(i,j),World)}$$

member3(I,World) becomes U if there is no information I. If negative information is found for I, member3(I,World) = U. The truth table for ~ was given in section 1.

To obtain negative information about constr-sem(i,j), the design of World is very important. There are probably many constraints between constr-sem(i,j) and the contents of World. We are currently designing World, but this is not discussed in this paper. We are ready to study *discourse*.

4. System for Dependency Grammar

This section briefly presents the basic grammar for the Japanese dependency structure.

In unification grammar formalisms, semantic features are propagated through constituent structure. In our system of dependency structure, semantic features are propagated through modifying relations. Feature propagation is written as shown in the following example. As stated above, Sem(i,j) ⇒ Syn(i,j),then Sem(i,j) and Syn(i,j) are written together.

```
mod ; Noun-with-agent-case,Verb : ↓ #{agent/←} = →
                  Sem(i,j)              Syn(i,j)
```

Semantic features propagated from the noun with the agent case are written with ←. Semantic features going from Verb to a *PHRASE* modified by the verb are written with →. ↓ denotes semantic features of the verb.

5. System for Semantics

The semantic framework is defined based on situation semantics [Barwise and Perry]. [Mukai 87] gives more details of this framework. For notational convention, a single colon is used in a line to represent repetition. Signs with brackets < > are terminology in the situation theory [Barwise 87].

```
5.1 <soa> ::= <state of affairs>
5.2 <state of affairs> ::=
          {sort/soa,
           relation/string,
           polarity/<polarity>,
           <argument place name>/<object>,
                :
           <argument place name>/<object>
          }
5.3 <property> ::=
          {sort/property,
          relation/<logical condition based on thesaurus entry>}.
5.4 <polarity> ::= 0 | 1 | UNBOUND.
5.5 <object> ::= <soa> | <parameter>
5.6 <parameter> ::=
          {sort/parm,
          name/string,
          anchor/<object>,
          property/<property>}
```

5.1 indicates that <soa> means a state of affairs in the situation theory.

5.2 shows the structure of <soa>. <soa> is constructed from one indicator and four kinds of constituents. sort/soa is an indicator, and indicates the type of this object. relation/<string> represents the relation of this <soa>. polarity/<polarity> indicates the polarity of this <soa>. <argument place name>/<object> indicates the arguments of <soa>. The number of arguments is ideally none to infinite .

5.3 indicates <property> as a restriction for <object> at <argument place name> in <soa>. <property> has <sort/property> as an indicator and <relation/..> as a restriction.

5.4 indicates polarity. The situation theory does not use *UNBOUND*. In this framework, *UNBOUND* is used to represent some contextual constraint in the lazy evaluation mechanism.

5.5 indicates the <object> which should be placed in the argument position of <soa>.

5.6 represents the structure of <parameter> which can be placed at the argument position of <soa>. <parameter> is one kind of minimum data in semantic analysis. <parameter> has one indicator, sort/parm , and three kinds of constituents. name/<string> is the name of a parameter. <parameter> has anchor/<object> as its anchor. There is probably some ambiguity when anchoring <parameter> to some <object> in context analysis, especially in discourse analysis.

6. Conclusion

As stated above, the logical approach to show and propagate constraints explicitly by three-value logic in the lazy evaluation mechanism opens the door to contextual analysis. Basically, this approach is backtrack free. However, some study about phenomena such as garden path sentences is necessary.

From the viewpoint of computer linguistics, backtracking is very expensive. The approach presented in this paper will achieve high performance in execution. Human beings do not backtrack like computers because there are very few people who can think without materials such as blackboards. However, this should be studied in the field of cognitive science in the future.

We are now researching rules to maintain coherency of discourse. Some other contextual aspects such as honorifics [Sugimura 86] should be taken into consideration when we design the whole model of discourse. We believe that the situation theory will be a powerful tool in building the model. It should be studied thoroughly in future research.

Acknowledgements

We would like to thank Dr. Uchida, Chief of the Second Research Laboratory at ICOT, and Mr.Yoshikawa, Assistant Chief of the Second Research Laboratory at ICOT, and Mr. Yokoi, Director of the Electric Dictionary Research Laboratory, for their encouragement and for allowing us ample time for this work.

References

[Barwise and Perry].Barwise, J. and Perry, J., Situations and Attitudes, (MIT Press,

Cambridge, 1983)
[Barwise 87] Barwise, J., Recent Developments in Situation Semantics, in : Nagao, M., (ed.), Language and Artificial Intelligence, (North-Holland, Amsterdam, 1987) pp. 387-399.
[Grosz 87] Grosz, B.J., The Structures of Discourse Structure, in : Nagao, M., (ed.), Language and Artificial Intelligence, (North-Holland, Amsterdam, 1987) pp. 3-32.
[Fenstad, Halvorsen, Langholm, and Benthem] Fenstad,J.,E., Halvorsen P.K., Langholm T., Benthem J., Equations, Schemata and Situations: A framework for linguistic semantics, CSLI Report No.CSLI-85-29, 1985
[Hashimoto 80] Hashimoto, S., Kokubunpou Taikeiron (Structure of Japanese), Hashimoto Shinkichi Hakase Chosakushuu, No.7., (Iwanami, Tokyo, 1980), (in Japanese)
[Kaplan & Bresnan] Kaplan, R., M., & Bresnan, J. Lexical -Functional Grammar: A Formal System for Grammatical Representation,
[Mukai 85] Mukai, K., Unification over Complex Indeterminates in Prolog, ICOT Technical Report TR113, (1985)
[Mukai 86] Mukai, K., Anadic Tuples in Prolog, in : Minker, J. (ed.), the Preprint of the Workshop on Foundation of Deductive Database and Logic Programming, Washington DC., (1986)
[Mukai 87] Mukai, K., A System of Logic Programming for Linguistic Analysis based on Situation Semantics, the Proceedings of the Workshop on Semantic Issues in Human and Computer Languages, Half Moon Bay, (1987)
[Nakamura 86] Nakamura, J., Solutions for Problems of MT Parser -- Methods used in Mu-Machine Translation Project, Coling '86 (1986) pp.133-135.
[Nitta 86] Nitta,Y., Idiosyncratic Gap, A Tough Problem to Structure-bound Machine Translation, Coling '86, (1986) pp.107-111.
[Ozeki 86] Ozeki, K., A Multi-Stage Decision Algorithm for Optimum Bunsetsu Sequence Selection from Bunsetsu Lattice, IECE, COMP86-47, (1986), (in Japanese)
[Pollard] Pollard, J., C., Generalized Phrase Structure Grammars, Head Grammars, and natural language, A dissertation submitted to the Department of Linguistics and the Committee on graduate studies of Stanford University in partial fulfillment of the requirements for the degree of Phd.84.
[Sugimura 86] Sugimura, R. , Japanese Honorifics and Situation Semantics, Coling '86 (1986) pp.507-510.
[Tujii and Nakamura and Nagao 84] Tsujii, J., Nakamura, J., and Nagao, M., Analysis Grammar of Japanese in the Mu-Project -- A Procedural Approach to Analysis Grammar, Coling '84 (1984) pp.267-274.
[Watanabe 81] Watanabe, M., Nihongo Koubunron (Theory of Japanese Phrase Structure), (Hakama shobou, Tokyo, 1981), (in Japanese)
[Yoshida 72] Yoshida, S., Syntax Analysis of Japanese Sentence Based on Kakariuke Relation between Two Bunsetsu, IECE, Vol. 55-D No. 4 (1972) pp.238-244.