How to Get Preferred Readings in Natural Language Analysis

Jun-ichi Tsujii, Yukiyoshi Muto
Yuuji Ikeda, Makoto Nagao
Dept. of Electrical Engineering,
Kyoto University,
Yoshida-honmachi, Sakyo, Kyoto, 606, JAPAN

Abstract

This paper describes a parsing program called KGW+p which is designed for integrating various sorts of knowledge to get most preferred structural descriptions of sentences. The system accepts not only a set of rules specifying constraints which any descriptions of sentences should satisfy, but also preferential rules which are utilized in selecting most preferred descriptions among possible ones. During the parsing process, the preferetial rules are utilized to select feasible parsing paths. Furthermore, KGW+p is complete in the sense that it can generate all possible structural descriptions of sentences if required. The descriptions are generated in a preferential order.

1 Introduction

One of the most crucial problems in natural language processing is how to conquer the problem of combinatorial explosions in sentence analysis[Ford82][Tsujii84][Hirsh84][Pereira85]. Linguistic constraints so far formulated by theoretical linguists are too weak to prevent many possible interpretations from being generated. They have concentrated only on formulating of *syntactic constrains*, which are obviously insufficient for selecting single interpretations of input sentences.

On the other hand, various methods have been proposed, by researchers in Artificial Intelligence and Computational Linguistics for eliminating possible interpretations by referring to other sorts of knowledge such as semantic, pragmatic ones, etc. However, these methods are not satisfactory either, because most of them presuppose very restricted subject fields and cannot deal with the openness, the essential property of natural languages. They also formulated semantic and pragmatic knowldge as *constraints* which interpretations should satisfy.

However, human readers utilize various cues as preferential. That is, there are many sorts of knowledge which seem to be better formalized in the form of rules for selecting feasible interpretations.

In ordinary reading situations, human readers cannot expect to have all information necessary for deciding interpretations. On the contrary, they would have only incomplete, partial knowledge about contexts and subject fields. Even so, they can easily fix single interpretations for given sentences. They might select one interpretation for 'I saw Mary with a telescope' based on semantic intemacy between 'to see' and 'a telescope'. They might also select one interpretation for 'John was given a book by his uncle'. Such selections cannot be explained by *constraint* rules, because other interpretations are also possible.

In this paper, we propose a new parser called KGW+p in which *constraint* rules and *preferential* rules can be separately desrcirbed in modular forms and integrated in the parsing process both effectively and efficiently [Ikeda86][Muto88].

KGW+p is implemented on Symbolics 3600 by using the favor system.

2 Organization of KGW+p

Fig.1 shows the organization of KGW+p. KGW+p consists of three separate components, the structure building component (SBC), the preference rating component (PRC), and the scheduler. The SBC accepts constriant rules in the form of CFG rules with feature augmentations, and applies them to generate syntactically possible structures of sentences. The rules the SBC accepts are the rules in unification grammars.

The SBC is a bottom-up and breadth-first parser with top-down filtering, and constructs partial parse trees (PPTs) from left to right. Without the PRC and the scheduler, the SBC produces all syntactic structures compatible with a given set of constraint rules.

On the other hand, the PRC computes plausibility values (PVs) of PPTs generated by the SBC, and the scheduler loosely controls the whole parsing process based on PVs. The scheduler suspends less preferred parsing paths, and resumes them if the preferred ones go to deadend in the later stages of processing. Though KGW+p works at promissing parsing paths first, it can gererate, if required, all structural descriptions by resuming suspended paths.

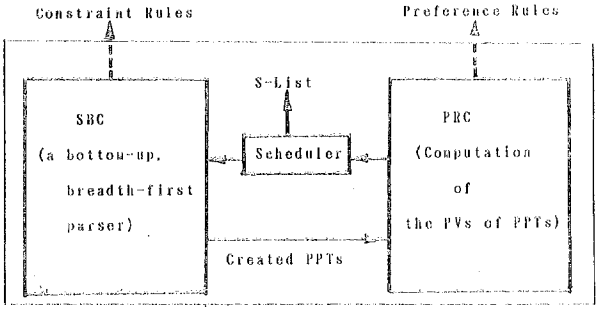

Fig. 1 The Organization of KGW+p

3 Algorithm of the SBC

Maximal flexibility in controlling CFG parsing can be obtained in the active chart parser [Kay80][Winograd83]. In this algorithm schemata, a parsing process is taken as a sequence of attaching an active or inactive arc to *Chart*, one at each time.

Though each attachment of an arc creates a set of arcs to be attached, the chart parser in a general form does not attach

them immediately, but registers them in *Agenda* . A scheduler decides which arc in *Agenda* is to be attched next. Because no *a priori* ordering of arc attachment is assumed, one can realize abitrary, flexible control mechanisms in this schemata.

However, such a maximal flexibility is obtained at the cost of efficiency. The scheduler has to be invoked at each cycle to decide which arc to be attached. Furthermore, when arcs are created, we have to take away the arcs which exist in *Agenda* or in *Chart*, before registering them in *Agenda*. The same arcs may exist to the arcs only whose leftmost constituents are filled by inactive arcs. Because of *the reachability condition*, we also have to check applicability of rules to all the inactive arcs in the right neighbor, whenever an active arc is attached. Such repeated checkings can be avoided in more restricted algorithms.

In KGW+p, we use an *Agenda*-like list (S-list - suspending list) but unlike the *Agenda* in the active chart parser, it only keeps the arcs which will be tried after the preferred ones fail. The other created arcs are attached immediately. Instead of the scheduler, the SBC has its own control scheme for building PPTs mechanically from left to right. The scheduler of KGW+p is more like a *demon* watching the SBC. When it finds less preferred PPTs (arcs) generated, it jumps out to store them in the S-list. Or when it finds the SBC goes to deadend, it decides which suspended PPTs in the S-list should be resumed.

The SBC in KGW+p uses two data structures, one for inactive arcs and the other for active arcs in *Chart*. As in Fig. 2, the inactive arcs from vertex i to j are stored in P(i, j) and the active arcs with ending vertices i are stored in G(i). We call the arcs in P(i, j) *inactive PPTs* and the arcs in G(i) *active PPTs*. Both P(i, j) and G(i) are realized as flavor instances of each type (P-Flavor and G-Flavor).

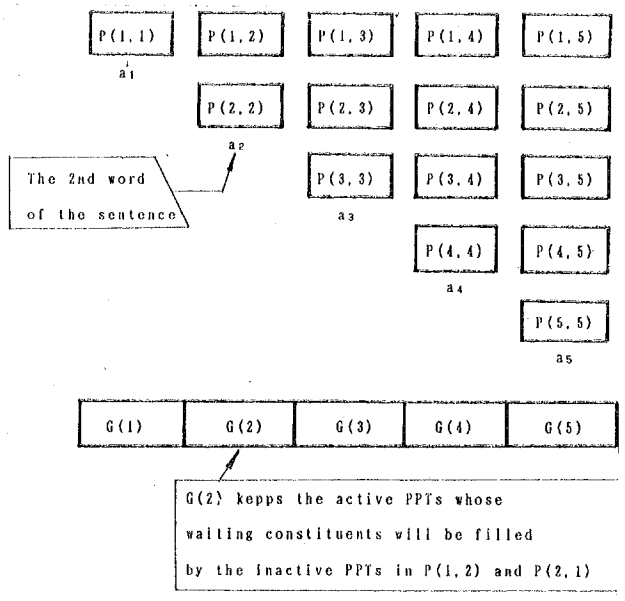

Fig. 2 Two Data Structures in KGW+p

We also realize active and inactive PPTs as instances of the PPT-flavor, each of which keeps the following items (c-PPT in the following means the PPT which is expressed by the flavor instance)

(1)starting and ending vertices of c-PPT
(2)syntactic category and features of the top node of c-PPT
(3)completed constituents: a list of inactive PPT-instances filling the child nodes of c-PPT
(4)remainders: a list of constituents to be filled. If c-PPT is inactive, the *remainders* is an empty list. We call the leftmost constituent of the remainders *the waiting constituent*. Note that the waiting constituents of PPTs in G(i) are to be filled by inactive PPTs with starting vertex i.
(5)pairs of a larger PPT which incorporates c-PPT as the leftmost constituent and the rule which was used to create the larger PPT.

Because PPT-instances keep (1) - (4) as the arcs in *Chart*, they can be suspended in the S-list (not immediately stored in P(i, j) or G(i)) and resumed afterwards freely. (5) is used to avoid redundant processings in the retrial phases (*see Section 4*).

The basic cycle of parsing is implemented as a set of methods of P-Flavor. When P(i, j)-instance is triggered, the methods in P-Flavor perform the following operations for each PPT stored in P(i ,j).

- (*Extension of Active PPTs*)look for active PPTs in G(i) which can incorporate the PPT as the leftmost constituent of the remainders, and create new PPTs
- (*New Rule Application*)look for rules whose leftmost constituents in *rhs* can be unified with the PPT and whose nonterminals in *lhs* can reach to the nonterminals of the waiting constituents of active PPTs in G(i), and applies them to create new PPTs

By storing newly created PPTs in the corresponding P(i', j') or G(j') immediately , a naive bottom-up, breadth-first and left-to-right parsing with top-down filtering can be easily realized as follows.

(1)After completion of the basic cycle, P(i, j)-instance triggers the execution of the basic cycle in P(i-*1*, j)-instance
(2)A trigger to P(*0*, j)-instance is taken as a trigger to the SBC-manager. The manager creates new PPTs by using the rules $A \rightarrow a_{j+1}$ (a_{j+1} is the j+1 th words), stores them in P(j+*1*, j+*1*) and triggers P(j+*1*, j+*1*)-instance
(3)Parsing is started by triggering P(*0*, *0*)-intance (This leads to the triggering of P(*1*, *1*) in (2)).

The basic control scheme of the SBC is the same as the above one. However, in KGW+p, after each basic cycle of creating new PPTs, newly created PPTs are rated by the PRC and the scheduler suspends less preferred (active or inactive) PPTs by storing them in the S-list. Only preferred ones are stored in corresponding P(i', j') or G(j') in parallel. Thus, though the scheduler loosely controls the whole process, the SBC analyzes sentences basically from left to right in a breadth-first manner by its own efficient control scheme. Note that the basic control scheme is an extended one of the algorithm proposed by V. Pratt [Pratt75] to deal with *n-ary* rules.

4 Resuming the Suspended PPTs

When all of preferred paths fail, the scheduler resumes some of suspended PPTs. This can be done simply by tranferring them from the S-list to the corresponding $P(i, j)$ or $G(j)$ and triggering $P(j', j')$. Here, j' is the smallest one among j of $P(i, j)$ and $G(j)$ in which the resumed PPTs are transferred.

After restoring the suspended PPTs, the same bottom-up, left-to-right and breadth-first parsing is performed from the *j'-th* word. However, special care is taken in KGW+p to reuse PPTs already constructed in the preceding trials to avoid duplicated processings.

We can reduce necessary processings in the *n-th* retrial phase as follows.

> (*Case-1*) $P(i, j)$ contains no PPTs newly creadted in the *n-th* retrial phase, and $G(i)$ contains no active PPTs newly created in the *n-th* retrial phase : we can completely skip the basic cycle for $P(i, j)$.
> (*Case-2*) $P(i, j)$ contains no PPTs created in the n-th retrial phase, but $G(i)$ has active PPTs newly created in the *n-th* retrial phase:
> While we have to perform the *Extension of Active PPTs* operation in the basic cycle for each PPT in $P(i, j)$, we only have to consider the new active PPTs in $G(i)$. This operation may lead to creation of new PPTs in the *n-th* retrial phase.
> We also have to perform the *New Rule Application* operation of the basic cycle, because the reachability condition may change. However, it may happen that the same rules have already applied to the PPTs in the former trials. In this case, because each PPT keeps a list of pairs of the larger PPTs and the rules (*see Section 3*), we can reuse the larger PPTs and avoid creating new PPTs in the *n-th* retrial phase.

In order to minimize the redundant processings in the retrial phases, P- and G-flavors provide different slots for PPTs created in the *n-th* trial and for those created in the former trials (*see Fig.3*). The analysis proceeds in the retrial phases in exactly the same way as in the first trail, but the duplication of operations are carefully avoided.

```
(1) P-Flavor
      new-ppts: keeps the inactive PPT-instances created
                in the n-th retrial phase.
      re-ppts:  keeps the inactive PPT-instances which
                are grown in the n-th retrial phase. But
                the same PPT-instances have been created
                in the former trial phases.
      old-ppts: keeps the inactive PPT-instances created
                in the former trial phases.
(2) G-Flavor
      new-ppts: keeps the active PPT-instances created
                in the n-th retrial phase.
      old-ppts: keeps the active PPT-instances created
                in the former trial phases.
```

Fig.3 Internal Structures of the P-Flavor and the G-Flavor

5 Format of Preference Rules and the PV Calculation

In the basic cycle, for each PPT in $P(i, j)$, the SBC creates a set of new PPTs which incorporate the PPT. These new PPTs represent different hypotheses based on the same bottom-up evidence, the incorporated PPT.

The PRC computes the PVs (plausibility values) of these different hypotheses by invoking a package of preference rules. A rule package is defined for computing the PVs of larger PPTs incorprating the same inactive PPT. That is, a rule package is defined for each syntactic category (nonterminal) of a PPT to be incorporated. A set of rules for PP-attachment, for example, are defined in a package which is invoked when the incorporated PPT is a prepositional phrase.

In order to compute PVs, we can refer in preference rules to various sorts of information as follows (we use *tree* and *TREE* for the incorporated PPT and the incorprating PPT, respectively).

> (1)the top node of *tree*
> (2)the top node of *TREE*
> (3)constituents of *TREE* already incorporated (the left brothers of *tree* in *TREE*)
> (4)sequence of active PPTs which eventually predict *TREE* (note that each PPT keeps the larger PPTs which will incorprate the PPT when it is completed - *see Section 3*)
> (5)lexical information of words which appear in the right unanalyzed portion of sentences (look-ahead)

(4) and (5) indicate the global nature of preference rules of KGW+p in the sense that the PVs of *TREEs* are computed not only from the constituents of *TREEs* but also from their surrounding contexts (Fig.4).

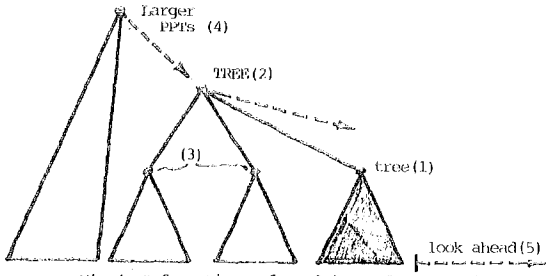

Fig.4 Information referred in Preference Rules

Fig.5 shows the format of preference rules. [Incompatible-Cases] enumerates different relationships between *tree* and *TREE*. In the package for PP-attachment, we enumerate as incompatible cases different types of PP-attachments such as PPs filling one of the valences of verbs, PPs as adjuncts, etc. A set of [Independent-Evidences] is defined for each incompatible case.

When a set of created PPTs with the same incorporated PPT are given, the PRC invokes a package, and for each created PPT, it determines which *exclusive case* matches with it. Then, the set of *independent evidences* for the case is evaluated.

Each independent evidence is expressed in a condition-value pair and, if the condition matches with the created PPT, it returns the specified value. The PRC gets a set of values from the independent evidences, each of which is a primitive PV based on a certain aspect of the PPT such as semantic intimacy of words, well-formedness of syntactic trees, etc. By combining the values with a certain function (currently, we use simple addition as this

function), the PRC determines the PV for the incorporation of *tree* into *TREE*.

```
(PRULE
    :CAT    (one of the syntactic Categories)
    :TYPE   (non-head, head) Depending on whether the category
                  is the maximal bar-level (head) or not.
:INCOMPATIBLE-CASES
    (:tree-cond ((condition of case1))
        :independent-evidences
            ((condition of evidence1) (value))
            ((condition of evidence2) (value))
            ((condition of evidence3) (value))
                  ..........
    (:tree-cond ((condition of case2))
        :independent-evidences
                  ..........
```

Fig. 5 Format of the Preference Rules

The actual PV of the created PPT is deterimined by the combination of

(1) PV in the above which is given to the combination of *tree* and *TREE*
(2) PV of *tree*
(3) PV of *TREE* : *TREE* has already incorporated left constituents and have accumulated their PVs
(4) PV of the larger PPTs which incorporate *TREE* when it is completed

Though one can consider any functions for integrating the above set of PVs, we use simple addition in the present experiments. And we do not use (4) (the PV from top-down) in this addition.

In the present experiment, after the PRC computes the PVs of larger PPTs incorporating the same PPT, the scheduler suspends PPTs which have low PVs compared with the most preferable PPT. That is, if the difference between the highest PV and the PV of a PPT exceeds a certain (predetermined) threshold, the PRC suspends the PPT.

6 Experiments

We conducted various experiments by using KGW+p. In this section, we will show the experiment of disambiguating sentences containing the word *that*. *That* can be taken as a pronoun, a determinner, a relative pronoun, a complementizer, a noun as an antecedent of a relative clause, a conjunction for an appositional clause or adverbial clause, an adverb, etc.

The followings are examples we realize as preference rules in the experiment. (Note that, in the present experiment, the PVs given by independent evidences are classified into 5 ranks, most preferred (+2), more preferred (+1), neutral (0), less preferred (-1) and least preferred (-2)).

```
(prule
  :cat   that
  :type  head
  :incompatible-cases ((incomp-case
                        :tree-cond   (:node-test (= m.cat  %*that))
                        :exist-goal  (:cg (:cat  %thatc)
                                      :bg ((:cat  %np1)
                                           (:node-test (member abs self.nsem)))
                                      :mg (:cat  %np1))
                        :independent-evidences ((ind-evi :vtype sem
                                                         :value +1)))
                       (incomp-case
                        :tree-cond   (:node-test (= m.cat %*that))
                        :exist-goal  (:cg (:cat  %thatc)
                                      :bg ((:cat  %*v))
                                      :mg (:cat  %vp))
                        :independent-evidences ((ind-evi :vtype syn
                                                         :value +1)))
                       (incomp-case
                        :tree-cond   (:node-test (= m.cat %*THATa))
                        :exist-goal  (:cg (:cat  %advc)
                                      :bg ((:cat  %sdec)
                                           (:node-test (=  self.so  +)))
                                      :mg (:cat  %sdec))
                        :independent-evidences ((ind-evi :vtype prg
                                                         :value +1)))
                       (incomp-case
                        :tree-cond   (:node-test (= m.cat %*centdet))
                        :independent-evidences ((ind-evi
                                                 :vtype prg
                                                 :value -2)))
```

Fig. 6 Example of Preference Rules

(1) Nouns such as *fact*, *news*, etc. are often collocated with appositional clauses. When the head of a noun phrase preceding *that* is one of such nouns, the *appositional clause interpretation* is more preferred.
(2) When the verb in the sentence is one of the verbs subcategorized by *that-clause*, the *complementizer interpretation* is most preferrred.
(3) When the word *so* or *such* appears in the preceding part of the sentence, the *adverbial phrase interpretation* is most preferred.
(4) PP-attachments over clauses are less preferred.
(5) Omission of relative pronouns is less preferred.
(6) The *pronoun* and *determiner interpretation of that* are less preferred in written texts.
(7) Different usages of a verb have different preferences. The verb *to tell*, for example, has five usages, 'to tell *sth* to *sb*', 'to tell *sth*', 'to tell *sb sth*', 'to tell *sb that-cl*' and 'to tell'. The last usage (the intransitive usage) has the least preference.

etc.

An example of actual preference rules is given in Fig. 6. The sentences in the following are used in the experiment.

1. I told the fact that sulfuric acid dissolves the metal.
2. I told the man that sulfuric acid dissolves the metal.
3. I was so tired that I could not move.
4. I was so surprised at the fact that John told us.
5. I told the fact that sulfuric acid dissolves the metal to John.

For 1 and 2, the SBC generates seven descriptions as follows.

(a) $[_s$ ... $[_{vp}$ tell $[_{np}$the $[_{np1}$fact $[_{app\text{-}cl}$that sulfuric ..]]]]]
(b)$[_s$ ... $[_{vp}$tell $[_{np}$the fact] $[_{that\text{-}cl}$that sulfuric ...]]]
(c)$[_s$ ..$[_{vp}$tell $[_{np}$the fact $[_{rel\text{-}cl}[_{np}$that sulfuric] ..]] $[_{np}$the metal]]]
(d)$[_s$ ...$[_{vp}$tell $[_{np}$the $[_{np1}$fact $[_{app\text{-}cl}$ $[_{np}$that sulfuric] ..]]]]]
(e)$[_s$...$[_{vp}$ tell $[_{np}$ the fact] $[_{that\text{-}cl}$ $[_{np}$ that sulfuric] ...]]]
(f)$[_s$...$[_{vp}$tell $[_{np}$the fact $[_{rel\text{-}cl}$that sulfuric ..]]][$_{np}$the metal]]]
(g)$[_s$...$[_{vp}$ tell $[_{np}$ the fact $[_{app\text{-}cl}$that sulfuric ..]][$_{np}$the metal]]]

(c) - (g) are rated low because they contain less preferred constructions. For example, (c) contains the ommision of a relative pronoun, the determiner interpretation of *that*, a PP-attachment over a clause (the phrase *the metal*), etc. As the result, (c) becomes the least preferred one among the interpretations.

(a) and (b) are most preferable for 1 and 2, respectively. The PVs of (a) and (b) in these sentences differ by the semantic condition that the usage 'to tell *sb that-cl*' prefers *human* as *sb* and by the collocation condition that the noun *fact* is often collocated with an appositional that-clause but the noun *man* is not. For the sentences 1 and 2, KGW+p succeeds in integrating such different sorts of preferential cues to give the highest PVs to the interpretations most preferable for human readers. Furthermore, because the paths which lead to these interpretations have the highest PVs during the whole parsing process, any threshold values can be used for suspending less preferred interpretations.

KGW+p also produces the valid interpretation for the sentence 3 in a straightforward way, but it encounters certain difficulties in 4 and 5.

At the time when the word *that* is analyzed in the sentence 4, the relative pronoun interpretation, which leads to the valid analysis, has a lower PV than the other two interpretaions. The adverbial clause interpretation supported by the word *so*, and the appositional clause interpretation supported by the word *fact* have higher PVs. Therefore, if the threshold value is low, the valid interpretation is suspended. Furthermore, both interpretations succeed, though they contain such a semantically less preferred structure as $[_s\ [_{np}$ John$][_{vp}\ [_v$ tell$][_{np}$ us$]]]$ and the wohle interpretations are rated low.

In the sentence 5, because the interpretation most preferable to human readers contains a PP-attachment over a clause, it is rated less preferable than the one which contains $[_{np}$ the $[_{np}$ metal $[_{pp}$ to John$]]]$.

These examples, especially the sentence 4, show that we need a mechanism to notice that the selected parsing paths become less feasible (even though they do not fail) than the suspended paths. This mechanism requires a global method for comparing completely remote PPTs. We also have to devise a sophisticated method for deciding the threshold value appropriately.

Table 1 shows the effect of the threshold values in the analysis of the sententce 5. In the case when the threshold is 2, only a single analysis result is obtained at the first trial, but the result is not the most feasible one for human readers

Threshold Values	9	8	7	6	5	4	3	2
Time for Completing the 1st Trial(sec)	16.04	15.64	15.69	15.64	11.24	10.84	10.39	6.95
Time for the PV Computation	5.63	5.55	5.51	5.49	4.05	3.96	3.78	2.28
No. of Suspended PPTs	0	2	4	4	9	13	15	29
No. of Parse Trees in the 1st Trial	18	17	16	16	5	4	3	1

Table 1. The Effect of Different Threshold Values

7 Conclusions

In this paper, we describe the organization and the basic algorithm of KGW+p. KGW+p allows one to prepare knowledge for natural language parsing in two separate forms. One is for the *constraint* type of knowledge and the other is for the *preference* type of knowledge.

By using the *constraint* type of knowledge, the SBC (Structure Building Component) in KGW+p produces partial parse trees mechanically from left to right in a breadth-first manner. The scheduler, which is a kind of *demon* watching the SBC, loosely controls the whole parsing process by utilizing PVs (Plausibility Values) given by the PRC (Preference Rating Component). The PRC uses the *preference* type of knowledge to compute the PVs.

KGW+p prepares a framework in which we can obtain *the flexibility of control, the modurarity in knowledge preparation* , and *the efficiency and completeness of parsing* at the same time.

It is a very delicate and difficult problem to decide the actual PVs of interpretations and the threshold value for suspending PPTs. Because various different sorts of factors may contribute to the PVs with different strengths, we certainly have to combine conventional NLP techniques with appropriate statistical and stochastic models. We hope that KGW+p gives us a good starting point for such future researches.

References

[Ford82] Ford, M., Bresnan, J., Kaplan, R. : A Competence Based Theory of Syntactic Closure, in The Mental Representation of Grammatical Relations (ed: Bresnan), MIT Press, 1982.

[Hirst84] Hisrt, G.J. : Semantic Ambiguity against Ambiguity, phD thesis, Brown University, 1984.

[Ikeda86] Ikeda, Y., Tsujii, J., Nagao, M.: Unification based Grammar and its Control in Parsing, SIG on Communication and Natural Language Processing, JSECE, 1987 (in Japanese).

[Kay80] Kay, M.: Algorithm Schemata and Data Structures in Syntactic Processing, Technical Report CSL-80-12, Xerox PARC, 1986.

[Muto88] Muto, Y., Tsujii, J., Nagao, M.: Preference Rule and their Application Mechanism in KGW+p, SIG on Communication and Natural Language Processing, JSECE, 1988 (in Japanese).

[Pereira85] Pereira, F.: A New Characterization of Attachment Preferences, in Natural Language Parsing (eds: Dowty, Karttunen, Zwicky), Cambridge University Press, 1985.

[Pratt75] Pratt, V.R.: A Progress Report, Proc. of 4th IJCAI, 1975.

[Tsujii84] Tsujii, J., Nakamura, J., Nagao, M.: Analysis Grammar of Japanese in the MU Project, Proc. of Coling 84, Stanford, 1984.

[Winograd83] Winograd, T.: Language as a Cognitive Process, Addison-Wesley, 1983