UNIFICATION-BASED DEPENDENCY PARSING OF GOVERNOR-FINAL LANGUAGES

(Hyuk-Chul Kwon)

Dept. of Computer Science
College of Natural Science, Pusan National University
30 changjun-dong, Keumjung-ku, Pusan 609-735, Republic of Korea
Phone : 82-051-510-2218(office)
Phone : 82-051-556-6223(home)
E-mail : hckwon @ cosmos.kaist.ac.kr
Fax : 82-051-510-1792

(Aesun Yoon)

Dept. of French
College of Cultural Sciences, Pusan National University
30 changjun-dong, Keumjung-ku Pusan 609-735, Republic of Korea

ABSTRACT

This paper describes a unification-based dependency parsing method for governor-final languages. Our method can parse not only projective sentences but also non-projective sentences. The feature structures in the tradition of the unification-based formalism are used for writing dependency relations. We use a structure sharing and a local ambiguity packing to save storage.

This paper was supported in part by NON DIRECTED RESEARCH FUND, Korea Research Foundation, 1989

I. Introduction

The parsers of phrase structure grammars face troubles for parsing free word order languages in following respects.

First, they require a large size of grammatical rules for parsing free word order languages. *Second*, the free word order often results in discontinuous constituents(Covington, 1988). A phrase-structure tree of a sentence with discontinuous constituents would have crossing branches. This crossing branches can not be represented by conventional context free rules. *Third*, free word order languages feature very rich systems of morphological markings(Kwon, 1990). Word arrangements

and morphological markings are obviously contingent on relations between wordforms rather than on constituency(Mel'cuk, 1988).

One approach to parse free word order languages is the principle-based parsing(Berwick, 1987). The other approach is the dependency parsing(Mel'cuk, 1988).

This paper describes a unification-based dependency parsing method for governor-final(head-final) languages like Korean and Japanese. We develop the parsing method with special reference to Korean but the method can be adapted directly to Japanese parsing. Korean and Japanese are relatively free word order languages(Kwon, 1990). Although their word order is free except that dependents always precede their governor, word order variations lead to different emphasis on the topic and the focus. In contrast, their morpheme order is fixed at the level of words.

In Korean and Japanese, it is quite natural to drop any arguments including a subject and an object if they can be recovered through the context. Null subjects are also found in Italian and Spanish(Moon, 1989). Null arguments make it much harder to parse Korean and Japanese using phrase structure grammars. Because dependency grammars analyze syntactic structure as the relationships between ultimate syntactic units(i.e, morpheme, part of speech), dependency parsers can easily parse sentences with null arguments.

This paper follows the grammatical formalism of Mel'cuk(1988), but modifies it for computational efficiency and Korean specific characteristics. We try to parse not only projective sentences but also non-projective sentences. Feature structures in the tradition of unification-based grammars are used for writing dependency relations. But unification operation is modified for parsing non-projective sentences. A structure sharing and a local ambiguity packing is used to save storage.

II. Dependency Relations and Feature Structures

Mel'cuk differentiates three dependency relations : morphological dependency, syntactic dependency and semantic dependency(Mel'cuk, 1988).

The syntactic dependency is binary relations between wordforms, which are anti-symmetric, anti-reflexive and anti-transitive. The syntactic relations are represented by arcs : X -> Y: where X governs Y; X is called the governor of Y; and Y is called the dependent of X. The syntactic relations are best represented by a connected directed labeled graph.

Mel'cuk gives additional restrictions on the syntactic structure. First, a syntactic structure contains exactly one node(root) that does not depend on another node. Second, in a syntactic structure, no node may simultaneously depend on two or more other nodes. The syntactic structure becomes a rooted tree, specifically a D-tree by these two restrictions.

In Korean and Japanese, there are two different morphemes: free(content) morphemes and bound(function) morphemes. Bound morphemes include postpositions and verbal endings. A free morpheme can depend on another morpheme directly. But a bound morpheme can depend on another morpheme after it governs other morphemes. This means that the leaf nodes of the D-tree are always free morphemes.

We use feature structures in the tradition of unification-based grammars for writing dependency relations(Sells, 1985).

governor	relation	dependent
[cat : postposition]	case-marking	[cat : noun]
[cat : verb-stem]	actant	[cat : postposition attributive : - coordinative : -]
[cat : verbal-ending]	modal-marking	[cat : verb-stem]
[cat : noun]	attributive	[cat : postposition attributive : +]
[cat : noun]	coordinative	[cat : postposition coordinative : +]

< Table 1>

<Table 1> shows parts of the government pattern of Korean. As Korean and Japanese are governor-final languages, dependents always precede their governors. But there are no precedence relations between dependents in general.

[lex : "John"
cat : noun
animate : +]

[lex : "Susan"
cat : noun
animate : +]

[lex : "i"
cat : postposition
case : nominative
bound : +]

[lex : "ul"
cat : postposition
case : accusative
bound : +]

[lex : "po"
cat : verb-stem
{subcat=>subj,obj}
subj : [animate : +]]

[lex : " da"
cat : verbal-ending
modal : declarative
bound : +]

< Dictionary 1 >

<Dictionary 1 > is a sample Korean dictionary. The feature "bound" is used to differentiate between bound morphemes and free morphemes. When a bound morpheme governs another morpheme, the value of "bound" become "nil". As "bound" is not controlled by the unification operation, the change of the value of "bound" does not destroy the monotonicity of the unification. More explanation will be found in chapter III.

(1) John - i Susan - ul po - da
SM OM VS VE
(see) (DEC)

(2) Susan-ul John-i po-da

< SM : Subject Marker,OM :Object Marker, VS : Verb Stem, VE : Verbal Ending, DEC : DEClarative >

In (1) and (2), the subject marker("i") governs "John" and the object marker("ul") governs "Susan". "Po" governs both the nominative construction ("John-i") and the accusative construction ("Susan-ul"). Because of no dependency between "John-i" and "Susan-ul", there is no precedence relation between them. "da" governs "John-i Susan-ul po". As a result, both (1) and (2) are grammatical sentences and they have the same meaning as "John sees Susan".

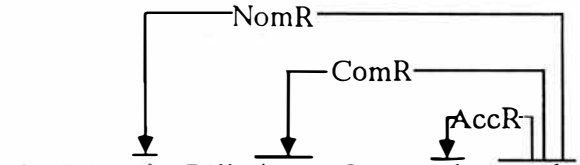
(3) John-i Bill-kwa Susan-ul po-da

Although "kwa" is a postposition, it can depend on a verb stem or a noun, but not both. When it depends on a verb stem, its meaning is "with". But its meaning is "and" if it depends on a noun. <Dictionary 2> shows the lexical information of "kwa".

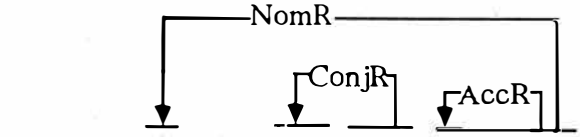
lex : "kwa" cat : postposition case : commitative bound : +	lex : "kwa" cat : postposition case : conjunctive coordinative : + bound : +	lex : "eui" cat : postposition case : possesive attributive : + bound : +

< Dictionary 2 >

From <Table 1> and <Dictionary 2>, we conclude that (3) has two different interpretations.



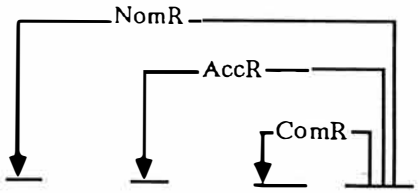
(3') John -i Bill -kwa Susan -ul po -da
(John sees Susan with Bill)



(3") John -i Bill -kwa Susan -ul po -da
(John sees Susan and Bill)

< NomR : Nominative Relation, AccR : Accusative R, ConjR :Conjunctive R, ComR :Comminative R>

But (4) has only one interpretation.

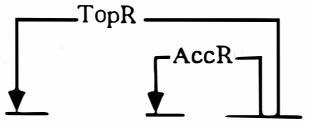


(4) John-i Susan-ul Bill-kwa po-da

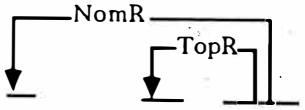
(John sees Susan with Bill)

<Table 1> and <Dictionary 2> also show that the possessive postposition "eui(of)" only depends on a noun.

The subcategorization of a verb gives additional constraints on the dependency relations. The subcategorization is used for a case assignment, the decision of null arguments and a filter on governing patterns. When a subject and an object are topicalized, the subject marker and the object marker are replaced to topic markers.



(5) John-un Susan-ul po-da
TM



(6) John-i Susan-un po-da
TM

< TM :Topic Marker, TopR : Topical R >

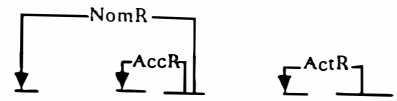
Postpositions do not provide the sufficient information for the case assignment of topicalized constructions in (5) and (6).

In (5), "po" governs the topicalized construction and the accusative construction. But verb stem "po" subcategorizes both a subject and an object. So, the noun of the topicalized construction is the subject of (5). (5) and (6) have the same meaning as (1) except that the subject and the object are topicalized respectively.

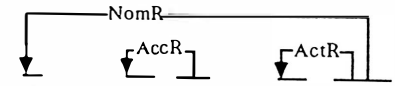
In Korean, the noun of a nominative construction is always the subject of a verb, and the noun of an accusative construction is the object of a verb, but not vice-versa. Therefore, we separate the case marking operation and the case assignment operation. The case of a topicalized construction is assigned when a verb stem is governed by a verbal ending.

(7) John-i Susan-ul po-ass - da - ko malha - da
VE VE VE VS VE
(past) (DEC) (COMP) (say) (DEC)
<COMP : COMPlementizer>

The decision of null arguments also requires the subcategorization. As the verb stem "malha" subcategorizes a subject and a complementizer("ko"), and "po" subcategorizes a subject and an object, two subjects are required in (7). But there is only one nominative construction. The nominative construction can be governed by "po" or "malha", but not both. As a result, we can conclude that one subject is dropped. (7) has two different interpretations as below.



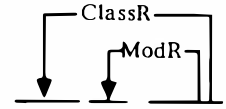
(7') John-i Susan-ul po - ass da-ko malha-da
(? says that John saw Susan)



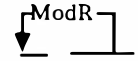
(7") John-i Susan-ul po-ass-da-ko malha - da
(John says that ? saw Susan)

< ? : null argument, ActR : Actant Relation >

Another constraints are required to parse the constructions with numerals of Korean and Japanese.



(8) i) sajen se kwon(three dictionaries)
NOUN DET NOUN
(dictionary)(three)Book.Form



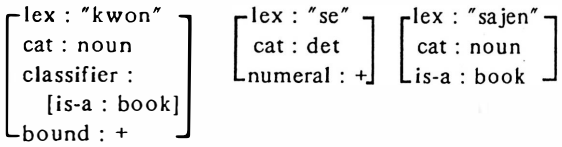
ii) se kwon (three book-like materials)
iii) *sajen kwon(not allowed)
iv) *se sajen kwon(not allowed)

<"kwon" : a unit for counting book-like materials, DET : determiner, ModR : Modificative Relation, ClassR : Classificative Relation>

"kwon" is a noun but a bound morpheme. We call it an incomplete noun. "kwon" can govern a numeral and a noun but there are restrictions in the governing order. "kwon" can govern a noun only after it governs a numeral, but the opposite is not true. This additional precedence restrictions can be formulated as <Table 2 > and <Dictionary 3>.

governor	relation	dependent
[cat :noun]	modificative	[cat:det]
[cat : noun, modifier [lex:det, numeral : +]]	classificative	[cat:noun]

< Table 2 >



< Dictionary 3 >

The second row of <Table 2> shows that a noun which is modified by a numeral (determiner) can govern a noun. The dictionary also shows that "kwon" is an incomplete noun and is a unit for counting books. There is a morphological dependency between "kwon" and "sajen". The above shows how our system deals with the morphological dependencies and additional precedence restrictions using feature structures.

III. Parsing Projective Sentences and Structure Sharing

Using dependencies for parsing natural languages, the projectivity is an extremely important property of the word order. A sentence is called projective if and only if the arcs of dependency links satisfy following restrictions(Mel'cuk, 1988).

(i) No arc crosses another arc

(ii) No arc covers the root of D-tree

Although most sentences of natural languages are projective, there exist several types of non-projective sentences. Non-projective sentences have discontinuous constituents. This chapter gives a parsing algorithm for projective sentences. The algorithm will be modified for non-projective sentences in the next chapter.

The algorithm scans a sentence from left-to-right for searching a governor. If a governor is found, it tries to make all the dependency links between the governor and the constructions whose head is the morpheme which immediately precedes the governor. The term *head* is used in the sense of top node of a construction as Mel'cuk(1988).

In a projective sentence, a governor can govern a wordform if and only if the governor governs directly or indirectly all the wordforms between them. Let $<m_1, m_2, \ldots, m_n>$ be an ordered list of morphemes. If m_i governs m_j and m_j governs m_k, then m_i indirectly governs m_k. The morpheme m_i can govern m_j if and only if all the morphemes between m_j and m_i are governed directly or indirectly by m_i where $j < i$. A head governs directly or indirectly all the other morphemes in a construction.

Our parsing strategy is as follows.

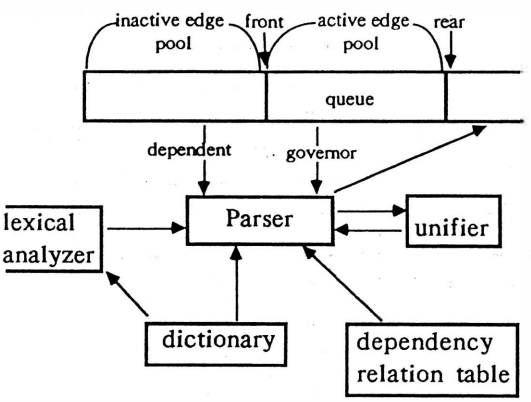
First) The parser gets a morpheme m_i from the lexical analyzer until an end-of-sentence marker is encounted.

Second) The parser searches constructions whose head is m_{i-1}. When there exist dependency relations between m_i and some of them, the parser generates new constructions and stores them in the queue.

Third) When some constructions exist in the queue, the parser gets one of them from the queue. Otherwise, goto *first)*. Let that construction contain all the morphemes from m_j to m_i where $j < i$ and m_i is its head. The parser searches construc-

tions whose head is m_{j-1}. When there exist dependency relations between m_i and some of them, the parser generates new constructions, stores them in the queue and repeats *third)*.

We implement the algorithm by chart. <Fig.1> shows the architecture of a Korean parser which runs at Apollo workstations.

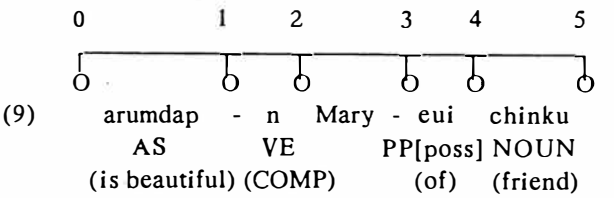


<Fig. 1>

The parser joins one dependent to one governor at a time. Each edge has a starting point and an ending point.

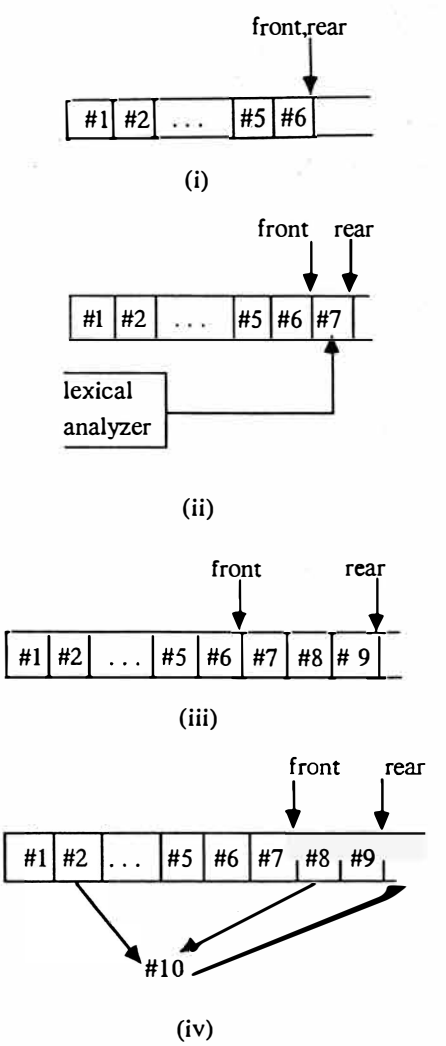
#	SP	EP	Construction	Remark
1	0	1	arumdap	
2	0	2	[arumdap,n]	
3	2	3	Mary	
4	0	3	[[arumdap,n],Mary]	beautiful Mary
5	0	4	[[[arumdap,n],Mary],eui]	of beautiful Mary
6	2	4	[Mary, eui]	of Mary
7	4	5	chinku	
8	2	5	[[Mary,eui],chinku]	friend of Mary
9	0	5	[[[[arumdap,n],Mary],eui],chinku]	friend of beautiful Mary
10	0	5	[[[arumdap,n],Mary,eui]],chinku]	beautiful friend of Mary

<Table 3>



<AS : Adjective Stem, VE : Verb Ending, PP[poss] : Possesive Postposition>

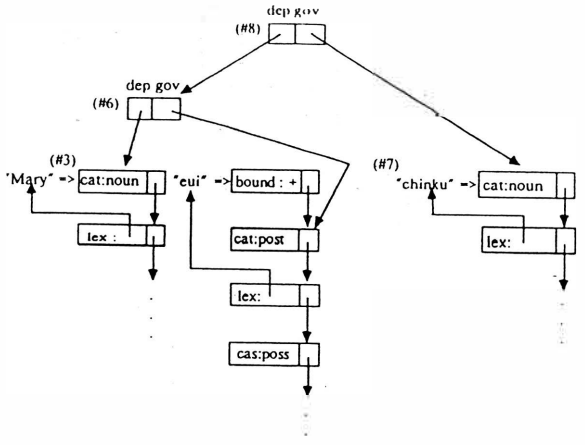
<Table 3> shows the content of the pool while (9) is parsed. (9) means "a/the friend of Mary who is beautiful" and has two different interpretations as (#9) and (#10). <Fig .2> shows the state of the pool when "chinku" is processed.



<Fig. 2> The state of the pool

(i) is the pool after processing "arumdap-n Mary-eui". As the inactive edge pool is empty, the parser gets "chinku" from the lexical analyzer as (ii). When the processing of (#7) is finished, the pool become (iii). (iv) shows the pool when (#10) is generated. As bound morphemes ("n","eui") can not depend on other morphemes by themselves, it is not necessary to store bound morphemes at the pool.

The storage for parsing grows exponentially as ambiguities are increased. We use a structure sharing(Tomita, 1986) and a local ambiguity packing(Shieber, 1986) to save storage. Although the order of the features is not important in the unification formalism, we always place the "bound" feature first.



<Fig. 3>

<Fig. 3> shows that (#8) shares the structures of (#6) and (#7). (#6) shares the structure of "eui" except for the "bound" feature. As the "bound" feature is excluded, the monotonicity of the unification is not destroyed.

We state that two or more subtrees represent a local ambiguity if they have the same starting point and the same ending point and if their top nodes are the same wordform. That is, (#9) and (#10) of the <Table 3> represent a local ambiguity. If a sentence has many local ambiguities, the total ambiguities would grow exponentially. To avoid this, we use a technique called local ambiguity packing which is suggested by Tomita(1986).

#	SP	EP	constructions
1	0	1	"arumdap"
2	0	2	[#1 | "n"]
3	2	3	"Mary"
4	0	3	[#2 | #3]
5	0	4	[#4 | "eui"]
6	2	4	[#3 | "eui"]
7	4	5	"chinku"
8	2	5	[#6 | #7]
9,10	0	5	[OR([#2,#6],#5)|#7]

[[#2,#6]|#7] = [#2|#8] = #10

<Table 4>

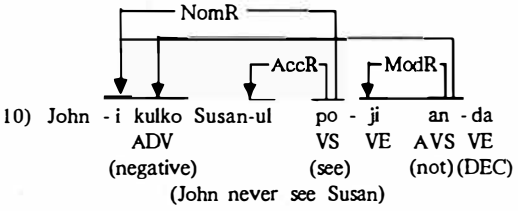
<Table 4> is the content of the pools after (9) is parsed with a structure sharing and a local ambiguity packing. #(9,10) in <Table 4> is the result of the local ambiguity packing of (#9) and (#10) in <Table 3>.

IV. Parsing Non-Projective Sentences

Non-projective sentences give serious difficulties in parsing natural languages.

But almost all languages have some sorts of non-projectivity(Mel'cuk, 1988).

There are two types of non-projectivity in Korean. The first one is related to the feature co-occurrence where the dependency links do not pass over the sentence boundary.



< *("kulko" ~ "an") = never, AVS:Auxiliary Verb Stem, ADV:ADVerb>

"kulko" is used only in negative sentences. In (10), "po" governs "John-i" and "Susan-ul", but the auxiliary verb stem "an" governs "kulko" and "po-ji". "kulko" can be placed anywhere before "an" at (10).

In a non-projective sentence, a governor can govern a wordform although the governor does not govern directly or indirectly some wordforms between them. This is one of the greatest obstacles for parsing non-projective sentences by our parsing method.

To overcome this problem, we introduce a new type feature called a co-occurrence feature. A co-occurrence feature-value is represented as ["fn" : c "v"], where "fn" is a feature name and "v" is a value. ["fn" : c "v"] means that its governor must have the feature-value ["fn" : "v"].

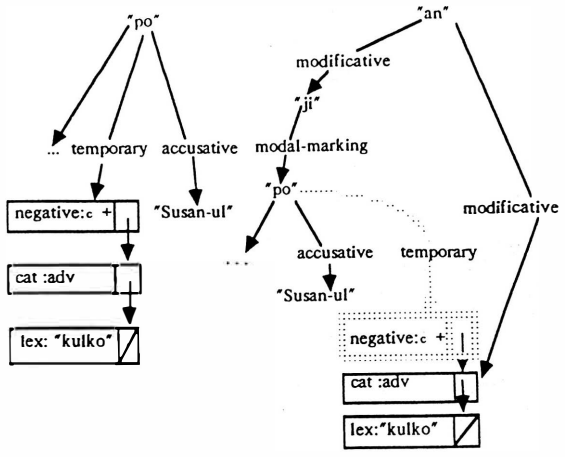
[negative : c +, cat : adverb, lex : "kulko"] [negative : +, cat : aux-verb-stem, lex : "an"]

<Dictionary 4>

governor	Relation	Dependent
[cat :verb-stem]	temporary	[negative : c +, cat :adverb]
[cat:aux-verb-stem, lex : "an"]	modificative	[negative : c +, cat :adverb]
[cat:aux-verb-stem]	modificative	[cat:verbal-ending]

<Table 5>

The first row of < Table 5 > shows that a verb stem temporarily governs an adverb which has [negative : c +]. When the verb stem depends on a construction which has [negative : +] and the dependency does not pass over the sentence boundary, the temporary dependency link is removed and a new dependency link between the adverb and the construction is connected. Two constructions are not unified when their dependency is temporary. We handle the co-occurrence feature similar to the "bound" feature.

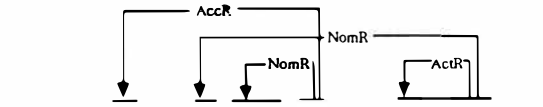


(i) (ii)

<Fig. 4>

When "an" governs "po-ji", a new link between "an" and "kulko" replaces the temporary link between "po" and "kulko". It is important that [negative: c +] is removed in (ii). If some co-occurrence features remain after the parsing, the sentence is incorrect.

The other type of non-projectivity occurs by non-local dependencies. Some constructions which are the dependents of an embedded verb can be placed at outer sentences in Korean. We can also find non-local dependencies in Finnish(Karttunen, 1986).



As stated above, "po" subcategorizes a subject and an object, and "malha" subcategorizes a subject and a complementizer. (11) has the cross arcs because "malha" governs "John-i" and "po" governs "Susan-ul".

Karttunen shows that this problem can be solved by functors with a floating type in Finish(Karttunen, 1986). The same framework also works in Korean. The framework can yield more than one results, but most of them are only acceptable at extraordinary situations. Therefore, our system strengthens the framework as a construction can be combined only with the nearest verb stem which can govern it when there is no projective governor of it.

V. Conclusion

We have shown a unification-based dependency parsing method for governor-final languages like Korean and Japanese. Feature structures in the tradition of unification-based grammars have been used for writing dependency relations. Our method can parse non-projective sentences as well as projecitve sentences.

We implement a Korean parser by the method presented in this paper using C language. The first version parser only used a structure sharing. But the current version uses a structure sharing and a local ambiguity packing. The local ambiguity packing saves about 35% of storage for parsing sample sentences.

More efficient structure sharing method and the dictionary structure are under study. We plan to use our method for parsing fixed word order languages.

Reference

[1] Berwick, R. C. 1987 Principle-Based Parsing, A.I. T.R. No. 972, MIT AI Lab.

[2] Covington, M. A. 1988 Parsing Variable Word Order Language with Unification-Based Dependency Grammar, ACMC Research Report 01-0022, University of Georgia.

[3] Hellwig, P. 1986 Dependency Unification Grammar, Proc. of Colling 86, pp.195-198.

[4] Karttunen, L. 1989 Radical Lexicalism, Alternative Conceptions of Phrase Structure, The University of Chicago Press., pp44-65.

[5] Kashket, M. B. 1987 A Government-Binding Based Parser for Warlpiri, TR 993, MIT AI Lab.

[6] Kwon, H., Yoon, A., Kim, Y. 1990 A Korean Analysis System Based on Unification and Chart, Proc. of Pacific Rim International Conference on Artificial Intelligence '90, pp251-256.

[7] Mel'cuk, I.A. 1988 Dependency Syntax : Theory and Practice, State of University of New York Press.

[8] Moon, G. 1989 Ph.D.Diss.,The Syntax of Null Arguments with Special Reference to Korean, The University of Texas at Austin.

[9] Sato, P. T. 1988 " A Common Parsing Scheme for Left- and Right-Branching Languages," J. of Computational Linguistics, Vol. 14, No.1, pp.20-30.

[10] Sells, P. 1985 Lectures on Contemporary Syntactic Theories, CSLI.

[11] Shieber,S.M et al. 1986 A Compilation of Papers on Unification-Based Grammar Formalisms : Part II, Report No. CSLI-86-48

[12] Tomita, M. 1986 Efficient Parsing for Natural Language, Kluwer Academic Pub.