A Korean Parser with Unification-based Dependency Grammar*

Yeoung-Uk Park, Hyuk-Chul Kwon and Aesun Yoon

This paper presents a Korean parser with unification-based dependency grammar. The basic goal in implementing the parser is to provide an adequate method to deal with word order variability and ellipsis of Korean. Dependency grammar is used to achieve this goal and a parsing technique based on chart is suggested for parsing of governor-final languages. In the implementation of Korean parser, structure sharing and local ambiguity packing are used for computational efficiency.

1. Introduction

Korean is a relatively free word order language and has very rich systems of morphological markings. In Korean, it is quite general to omit obligatory cases including a subject or an object if they can be recoverable from the context. These characteristics make it difficult to parse Korean by phrase structure grammars.

Phrase structure grammars concentrate on constituency. Under the PS-approach, an actual sentence is cut into major constituents, each of which is subsequently cut in its turn. But constituency can not directly reflect the role of morphological markings and the word order variation. Ellipsis also makes it hard to parse Korean because there is no beginning-of-clause marker.

In recent work, Jung (1987), Yoon (1989) and Kwon (1990) have developed Korean parsers based on phrase structure grammars. Jung (1987) and Kwon (1990) use binary grammars based on HPSG. But their parsers require additional mechanisms for case assignment, and the role of the phrase structure rules can be negligible. Yoon's parser, based on LFG, ana-

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

lyzes the roles of the morphological markings while constructing f-structures.

This paper presents a Korean parser with unification-based dependency grammar. The dependency grammar concentrates on the relationships between ultimate syntactic units. The main logical operation here is the establishing of binary relations. The binary relations have some similarities with the binary grammars of the Jung (1987) and Kwon (1990). But the binary relations of the dependency grammar show *which terms are related to which other items and in what way*, while a pure phrase structure grammar does not allow for its representation. This makes it simple to analyze the roles of the morphological markings and to parse free word order languages. The dependency grammar also makes it easy to parse sentences with ellipsis because the dependency relations represent the relations between wordforms.

2. Syntactic Dependency of Korean

We will start with a review of some facts about Korean (Kwon & Yoon (1990)).

i) Korean is an agglutinative language in which most grammatical functions are expressed by suffixes.

ii) Word order is relatively free while the morpheme order in a word has strong constraints. But this does not mean that Korean word order is completely free. In Korean, dependents alway precede their governor. In this sense, Korean is a governor-final language.

iii) It is quite natural to drop any argument including a subject and an object, if it can be recovered through the context.

According to Mel'čuk, syntactic structure has to be represented by binary relations between wordforms which are anti-symmetric, anti-reflexive and anti-transitive (Mel'čuk (1988)). The relations are called syntactic dependencies and represented by arcs: X→Y where Y depends on X or, conversely, that X governs Y:X is called the governor of Y and Y is the dependent of X. He also argues that every node, except the top node, must have just one governor and he calls it the principle of the uniqueness of the syntactic governor. As the dependency relations are the relations between wordforms, Korean sentences with ellipsis can be described more naturally by dependency grammar.

(1) John -i hakkyo -e ka -ss -ta -ko malha -ta.
Post N Post VS TN SE COMP VS SE
(SM) (school) (Loc) (go) (past) (DEC) (talk) (DEC)
(Post: Postposition, N: Noun, VS: Verb Stem, TN: Tense, SE: Sentential Ending, COMP: Complementizer, SM: Subject Marker, DEC: Declarative, Loc: Locative)

Each of the verb stems "ka-" and "malha-" subcategorizes one subject. But in (1), there is only one nominative construction that can be a subject. As a result, (1) can have two different interpretations. This phenomenon can not be described naturally by phrase structure grammars. Kwon (1990) abandons the principle of functional completeness of LFG and Jung (1987) uses null production rules. But these approaches require too much computational cost for parsing free word order languages where ellipsis is quite general.

As Korean is a governor-final language, "ka" and "malha" can govern the nominative construction "John-i". But the principle of the uniqueness of the syntactic governor prevents "John-i" from being the dependent of both "ka-" and "malha-". In consequence, (1) has two different interpretations where the subject of the "ka-" or that of "malha-" is omitted.

The dependency framework can also provide an appropriate description of the word order variability. 〈Table 1〉 shows the dependency relations of Korean between syntactic categories. The syntactic categories are treated as the feature value fairs in our system.

Governor	Dependent
noun	pre-noun, noun, postposition[POSS], ending[COMP]
postposition	postposition, noun, pronoun, adverb
ending	stem of verb, stem of adjective, ending
stem of verb	postposition, noun, adverb, ending[COMP]
stem of adj.	postposition, noun, adverb
adverb	adverb

〈Table 1〉

3. Parsing Projective Sentences

Using dependencies in the parsing led to the discovery of an extremely

important property of word order: so called projectivity (Convington (1988)). There exist several types of non-projective sentences in Korean, but all of them are somehow marked: emphatically or stylistically. In this paper, we try to parse only projective sentences.

A projective sentence has no cross arc of dependency link between its wordforms. In a projective sentence, a governor can govern a wordform if and only if the governor governs directly or indirectly all the wordforms between that wordform and it. Let $<m_1, m_2, \cdots, m_n>$ be an ordered list of morphemes. If m_i governs m_j and m_j governs m_k, then m_i governs indirectly m_k. But m_i can not directly govern m_k because dependency relations are anti-transitive. The morpheme m_i can govern m_j if and only if m_{i+1} to m_{j-1} are governed directly or indiectly by m_i where $j<i$.

Our parsing strategy is as follows.

First) The parser gets a morpheme m_i from the morpheme list.

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

Third) When some constructions exist in queue the parser gets one of them from queue. Otherwise, the parser repeats this process from the first step until an end-of-sentence marker is encountered. Let that construction cover m_j to m_i where $j<i$. The m_i is the governor of that construction because Korean is a governor-final language. The parser searches constructions which cover m_{j-1}. When there exist dependency relations between m_i and some of the constructions which cover m_{i-1}, the parser generates new constructions, stores them in queue and repeats the third step.

〈Fig. 1〉 shows the architecture of our parser.

Actually, the parser gets morphemes from the lexical analyzer. While m_i is processed, the inactive edge pool has edges generated from morphemes before m_i, and the active edge pool has newly generated edges where their governor is m_i and their dependents are constructions in the inactive edge pool. The active edge pool is implemented by queue. Each edge covers morphemes as chart parsing (Wiren (1988)) does. It is important that edges, whose arguments including a subject or an object are not filled, can also be dependents for parsing sentences with ellipsis. In LFG, this is impossible because of the principle of functional completeness.

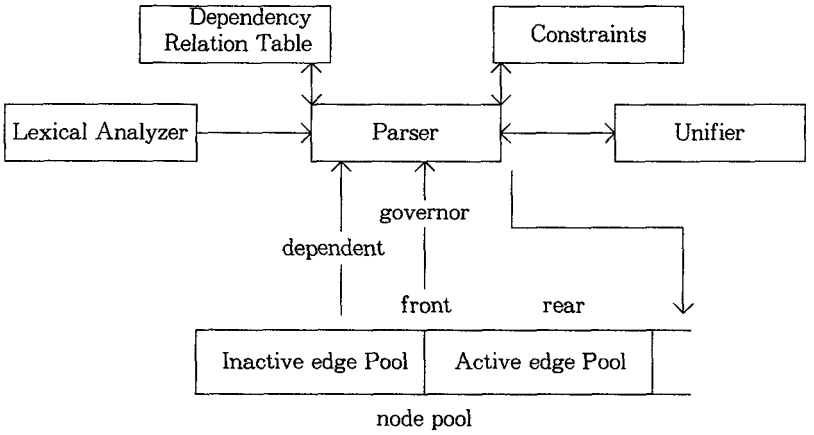


Fig. 1. The Architecture of the Parser.

〈Table 2〉 shows the state of the pool while "Susan-i arumdap-ta" is parsed.

(2) Susan -i arumdap -ta
Post AS SE
(SM) (is beautiful) (DEC)
(Susan is beautiful)
(AS: Adjective Stem)

inactive edge pool (dependent)	active edge pool (governor)
nil	(Susan)
(Susan)	(i), ((Susan), i)
(Susan), (i), ((Susan), i)	(arumdap), (((Susan), i), arumdap)
(Susan), (i), ((Susan), i), (arumdap), (((Susan), i), arumdap)	(ta), (((Susan), i), arumdap), ta) ((arumdap), ta)
(Susan), (i), ((Susan), i), (arumdap), (((Susan), i), arumdap), (ta) (((Susan), i), arumdap), ta) ((arumdap), ta)	nil

〈Table 2〉

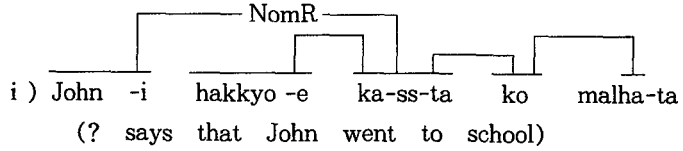
* $(D_1, D_2, \cdots, D_n, G)$: $n>=0$, $D_i(1<i<n)$: dependents, G: governor

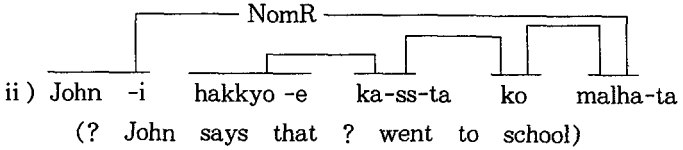
4. Parsing Sentences with Ellipsis

In Korean, any subcategorized argument can be omitted if it can be recoverable from the context. (1) in chapter 1 shows ambiguity caused by the ellipsis. As this ambiguity can not be disambiguated in the course of syntax analysis, our parser outputs all the parse trees from a sentence.

For parsing sentences with ellipsis, we allow that the constructions (edges) whose arguments are not filled can also be dependents. In LFG, this is impossible because of the principle of functional completeness. 〈Table 3〉 shows the state of the pool while (1) is parsed. It represents the process of parsing in word level. The first element of each active edge pool is the input from the lexical analyzer.

Although the subject of the third construction of [4'] in 〈Table 3〉 is not filled, it becomes the dependent of "malha" in the fourth construction of [5']. When parsing is over, we can find two constructions which cover the whole sentence.



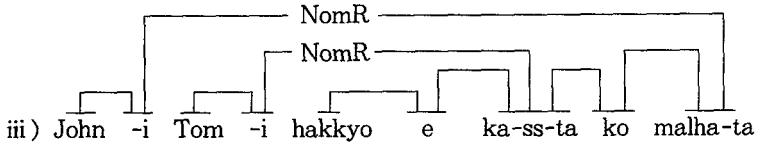


(NomR: Nominative Relation, ?: Omitted subject)

No	inactive edge pool(dependent)	No	active edge pool(governor)
0	nil	0	nil
1	nil	1'	(John-i)
2	{#1'}	2'	(hakkyo-e)
3	{#2} U {#2'}	3'	(ka-ss-ta), ((hakkyo-e), ka-ss-ta) ((john-i), ((hakkyo-e), ka-ss-ta))
4	{#3} U {#3'}	4'	(ko) (((john-i), ((hakkyo-e), ka-ss-ta)), ko) (((hakkyo-e), ka-ss-ta), ko), ((ka-ss-ta), ko)

5	{#4} U {#4'}	5'	(malha-ta), (((ka-ss-ta), ko), malha-ta) ((((hakkyo-e), ka-ss-ta), ko), malha-ta) ((((John-i), ((hakkyo-e), ka-ss-ta)), ko), malha-ta) ((john-i), ((((hakkyo-e), ka-ss-ta), ko), malha-ta))
6	{#5} U {#5'}	6'	nil

〈Table 3〉



i) and ii) depict the two constructions where the subject of the outer sentence or that of the inner sentence is omitted respectively.

(3) John -i Tom -i hakkyo-e ka -ss -ta -ko malha -ta
(John says that Tom went to school)

In (3), there is two nominative constructions. So, (3) has only one interpretation of iii) as the dependency links can not be crossed in a projective sentence and a verb stem can subcategorize only one subject.

5. Structure Sharing and Local Ambiguity Packing

It is not enough to parse Korean sentences only by dependencies between syntactic categories because of ambiguities. To reduce ambiguities, we use the feature structure of standard unification based grammars.

When there exists dependency between two constructions, the unifier unifies their features to check constraints and to build a representation of the result. But, it requires too much time and space if unification is always preceded by a copying operation. Therefore, we use structure-sharing technique (Shieber et el. (1986)). The structure sharing dramatically reduces time and space for parsing. The structure sharing method was introduced by Boyer and Moore and it is used in the implementation of PART-II (Shieber et al. (1986), Boyer & Moore (1972)).

It is desirable for practical natural language parsers to produce all possi-

ble parses and to store them for later disambiguation. The ambiguity of a sentence grows exponentially as the length of a sentence grows. Thus, the parser would take exponential time and space. Therefore, Tomita (1986) provides an efficient representation called local ambiguity packing so that the size of the parse trees does not grow exponentially.

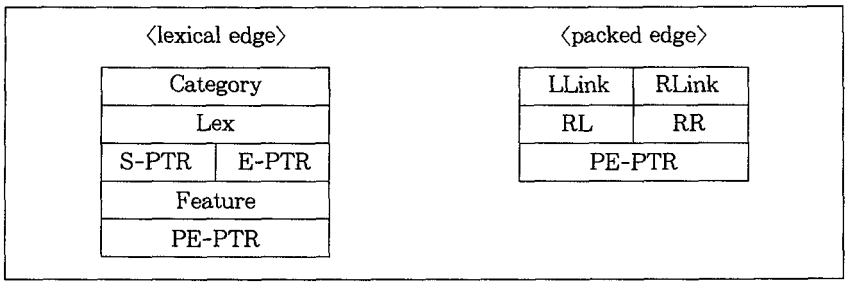
Two or more subtrees represent local ambiguity if they cover the same wordforms and their governors or their last wordforms have the same category value. The governor of a construction is always the last wordform because Korean is a governor-final language. The governors of subtrees that represent local ambiguity are merged and treated as if there were only one edge (Tomita (1986)). Such an edge is called a packed edge.

〈Table 4〉 shows the number of edges required for parsing a sentence of governor-final languages in worst cases. Although this type of sentence is quite rare in actual sentences, it is worth applying local ambiguity packing for parsing actual sentences.

# of morpheme	Local Ambiguity Packing	
	NO	YES
3	7	7
4	16	14
5	39	25
6	104	41
7	301	63
8	927	92
9	2983	129
10	9901	179
11	33615	231

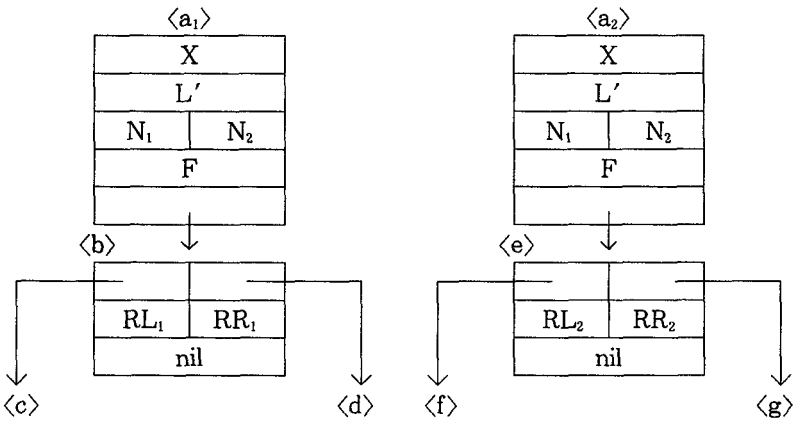
〈Table 4〉

Our system represents the syntactic dependency by binary tree. There are two different types of edge in our parser as 〈Fig. 3〉. A lexical edge represents the information of a lexical entry and a packed edge represents the dependency relation. PE-PTR of a packed edge points another packed edge when two locally ambiguous trees are merged. LLink and RLink of a packed edge point the left subtree and the right subtree respectively.



〈Fig. 3〉

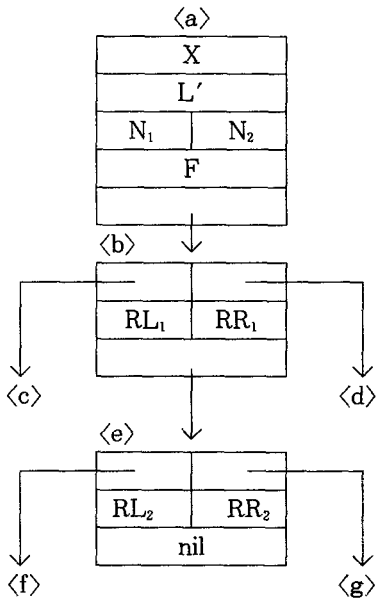
We represent the dependency relation by binary tree as follows. The left subtree 〈c〉 in 〈Fig. 4〉 is the dependent of 〈a_1〉 while the governor of the right subtree 〈d〉 is the governor of the 〈a_1〉. Therefore, the root of a dependency tree has only the left subtree because it does not depend on another construction. RL_1 represents the syntactic relation between 〈a_1〉 and the construction pointed by 〈c〉. The syntactic relation between the governor of <a_1> and the construction pointed by <d> is in RR_1.



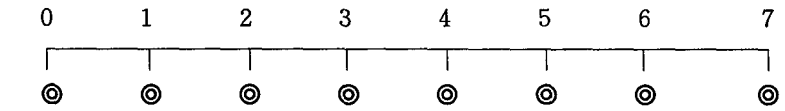
〈Fig. 4〉 Unpacked Trees.

The two trees of 〈Fig. 4〉 show local ambiguity because both of them cover the wordforms from N_1 to N_2 and their categories are the same. The result of local ambiguity packing of the two trees is in 〈Fig. 5〉.

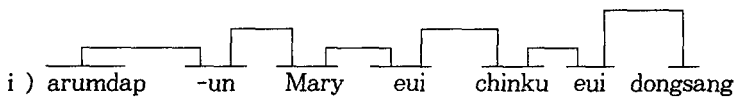
The construction (4) has three different interpretations as follows.



〈Fig. 5〉 Packed Tree.

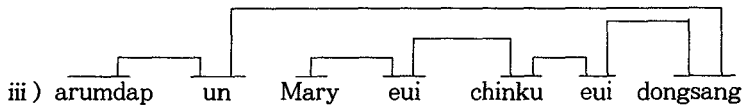


(4) arumdap -un Mary -eui chinku -eui dongsang
AS VE PP[poss] N N
(is beautiful) (Ad) (of) (friend) (sister)
(the sister of the friend of Mary who is beautiful)
(AS: Adjective Stem, PP[poss]: Possessive Postposition,
Ad: Adnominalizer, VE: Verb Ending)



(Mary is beautiful)

(The friend is beautiful)



(The sister is beautiful)

〈Fig. 6〉 in the appendix shows the result of the parsing of (4) where three different parse trees are packed into one. Some information of the edges is abbreviated in it. Both (j) and (g) represent two way local ambiguity because they have two packed edges. As one of the two packed edges of (j) governs (g), we can see three different parse trees in 〈Fig. 6〉.

〈Table 5〉 shows the empirical results of parsing five Korean sentences. This gives the necessity of local ambiguity packing and feature unification in parsing.

Examples

(1) chage -un nalssi -ka mul -ul eolum -euro mandu -nta.
AS VE N Post N Post N Post VS SE
(cold) (Ad) (weather) (SM) (water) (AM) (ice) (GOAL) (make) (DEC)

(2) John -i arumdap -un Mary -eui dongsang -ul joaha -nta.
AS VE PP[poss] N VS
(is beautiful) (Ad) (of) (sister) (like)

(3) John-i Seoul -lo nalaga -un bihangi -ul po -nta.
Post VS VE N VS
(GOAL) (fly) (Ad) (plane) (see)

(4) jijungha-n batch-file-i iss -nun disk -ka wonrea jijungha -n
VS N AS VE N Post ADV VS VE
(assign) (exist) (Ad) (TM) (originally) (assign) (Ad)
drive -e up -ta.
N Post AS SE
(Loc) (not exist) (DEC)

(5) myeongryeong -i teugjungha-n jagup -ul ha -nun bangbub-ul hangul
N VS N VS N N
(command) (assigned) (AM) (do) (method) (Korean)
MS-DOS -ege jisiha -nun jag -iun program -ita.
N Post VS AS VE N SE
(Dative) (direct) (small) (Ad) (DEC)

(ADV: Adverb, TM: Topic Marker, AM: Accusative Case Marker)

Local Ambiguity Packing	NO		YES		YES	
Feature Unification	NO		NO		YES	
example	# of parse trees	# of nodes	# of parse trees	# of nodes	# of parse trees	# of nodes
(1)	1	20	1	15	1	15
(2)	2	28	2	17	2	17
(3)	4	42	4	30	1	20
(4)	5	70	5	52	3	43
(5)	13	151	13	69	4	56

〈Table 5〉

6. Conclusion

This paper has given a Korean parser with unification-based dependency grammar. Korean is a free word order language where any subcategorized argument including a subject and an object can be omitted if it can be recoverable from the context. As dependency grammar concentrates on the relationships between ultimate syntactic units, we has been able to deal with word order variability in natural way and to parse sentences with ellipsis from their surface structures.

This paper has shown a parsing technique based on chart parsing for governor-final languages. For computational efficiency, we use structure sharing and local ambiguity packing. The empirical comparison shows the improvement of the space use by local ambiguity packing in parsing.

Our system can parse only projective sentences. Currently, we extend it to parse non-projective sentences. But this extension requires too much computational cost and causes it rise to generate too many parse trees from an embedded sentence. Therefore, we study on more efficient parsing technique and better disambiguation method.

References

Boyer, R. S. and J. S. Moore (1972) 'The Sharing of Structure in Theorem-proving Programs,' *Machine Intelligence* 7, pp. 101-116, John Wiley

and Sons, New York.

Covington, M. A. (1988) 'Parsing Variable Word Order Languages with Unification-based Dependency Grammar,' *ACMC Research Report* 01-0022, University of Georgia.

Hellwig, P. (1986) 'Dependency Unification Grammar,' *Colling '86*, pp. 195-198.

Jung, H. S. (1987) 'Korean Phrase Structure Grammar,' *Proc. of the First Natural Language Processing Workshop*, SIGAI of Korean Information System Society, pp. 3-37.

Kwon, H. C. and Aesun Yoon (1990) 'A Korean Analysis System Based on Unification and Chart,' *Proc. of Pacific Rim International Conference on AI. '90.*

Lee, B. C. (1990) *Theory and Practice of Dependency Grammar*, Chegi Pub., Korea.

Mel'čuk, I. A. (1988) *Dependency Syntax: Theory and Practice*, State University of New York Press.

Park, Y. U., H. G. Cho and H. C. Kwon (1990) 'Implementation of Korean Analysis System Using Dependency Grammar,' *'90 Spring Conference of Korean Information System Society*, Korea, pp. 191-194.

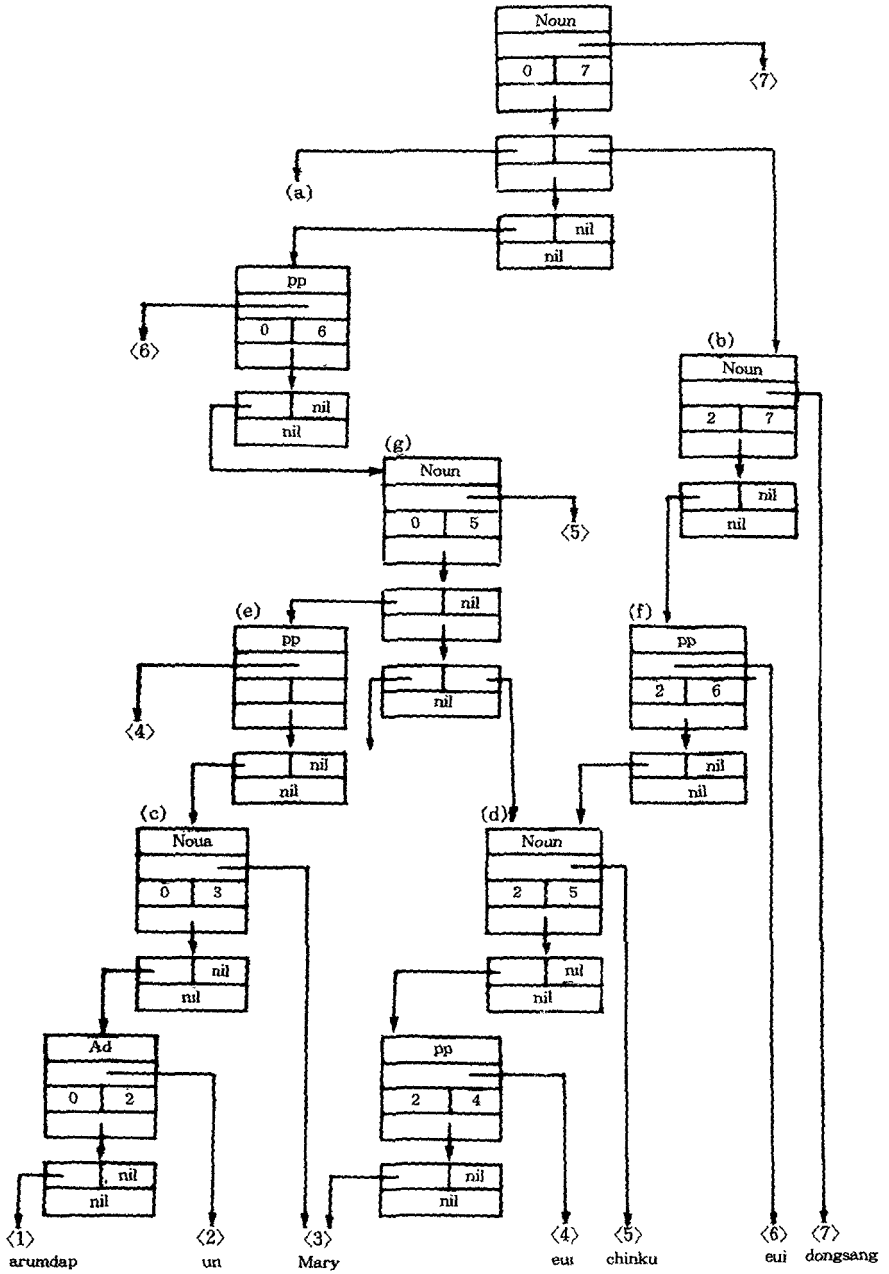
Shieber, S. M. et al. (1986) 'A Compilation of Paper on Unification-based Grammar Formalisms: Part II,' *Report No. CSLI-86-48.*

Tomita, M. (1986) *Efficient Parsing for Natural Language*, Kluwer Academic Publisher.

Wiren, M. (1988) 'On Control Strategies and Incrementality in Unification-based Chart Parsing,' *Thesis No. 140*, Linkoping University.

Yoon, D. H., Y. T. Kim (1989) 'Analysis Techniques for Korean Sentences Based on LFG,' *Proc. of International Workshop on Parsing Technologies*, pp. 369-378.

〈Appendix〉


Noun
0 7
〈7〉
(a)
nil
nil
pp
0 6
〈6〉
nil
nil
(b)
Noun
2 7
nil
nil
(g)
Noun
0 5
〈5〉
nil
nil
(e)
pp
〈4〉
nil
nil
(f)
pp
2 6
nil
nil
(c)
Noun
0 3
nil
nil
Ad
0 2
nil
nil
(d)
Noun
2 5
nil
nil
pp
2 4
nil
nil
〈1〉 arumdap
〈2〉 un
〈3〉 Mary
〈4〉 eui
〈5〉 chinku
〈6〉 eui
〈7〉 dongsang

Prof. Yeoung-Uk Park/Hyuk-Chul Kwon
Department of Computer Science
Pusan National University
30 Changjun-dong, Keumjung-ku
Pusan 609-735
Korea

Prof. Aesun Yoon
Department of French Language
Pusan National University
30 Changjun-dong, Keumjung-ku
Pusan 609-735
Korea