International Journal of Computer Processing of Oriental Languages
Vol. 16, No. 3 (2003) 197–218

World Scientific
www.worldscientific.com

Automatic Generation of Composite Labels Using Part-of-Speech Tags for Parsing Korean

SEONGYONG KIM* AND KEY-SUN CHOI†
Department of Computer Science,
Korea Advanced Institute of Science and Technology
373-1, Kusong-Dong, Yusong-Gu, Taejon, 305-701, Republic of Korea
*sykim@csone.kaist.ac.kr
†kschoi@cs.kaist.ac.kr

KONG JOO LEE
Department of Computer Science and Engineering, Ewha Woman's University
11-1, Daehyun-Dong, Seodaemun-Gu, Seoul, 120-750, Republic of Korea
kjlee007@ewha.ac.kr

We propose a format of a binary phrase structure grammar with composite labels. The grammar adopts binary rules so that the dependency between two sub-trees can be represented in the label of the tree. The label of a tree is composed of two attributes, each of which is extracted from each sub-tree, so that it can represent the compositional information of the tree. The composite label is generated from part-of-speech tags using an automatic labeling algorithm. Since the proposed rule description scheme is binary and uses only part-of-speech information, it can readily be used in dependency grammar and be applied to other languages as well.

In the best-1 context-free cross validation on 31,080 tree-tagged corpus, the labeled precision is 79.30%, which outperforms phrase structure grammar and dependency grammar by 5% and by 4%, respectively. It shows that the proposed rule description scheme is effective for parsing Korean.

Keywords: Syntactic Analysis; Korean; Agglutination; Binary Phrase Structure Grammar; Composite Label; Automatic Labeling Algorithm.

1. Introduction

Syntactic analysis of a language is the process of mapping the surface structures of a language into some tree-like structures. The basic task for the syntactic analysis is to define rules in a certain format with a general rule description scheme to cover the characteristics of the language. It is the goal of this paper

to propose a rule format with a simple algorithm to generate a rule label using only part-of-speech (POS) tag information automatically.

Korean is an agglutinative language. A sentence is made up of word phrases delimited by white spaces. A word phrase in turn is composed of one or more lexical morphemes concatenated by zero or more grammatical morphemes. Here, we need to choose a word phrase or a morpheme as the unit of rule description in order to describe a syntactic grammar for Korean. See the following sentence:

(1) (a) *sigan*/ncn[1] +*i*/jcs *gwijung*/ncps+*ha*/xsm+*n*/etm *geos*/nbn+*i*/jp+*da*/ef+./sf
(b) time+SUBJ[2] valuable+REL_CLS thing+be+DECL+
(c) Time is a valuable thing

Sentence (1) is made up of three word phrases, and nine morphemes including a sentential mark. The infix operator '+' stands for the concatenation, agglutination in other words, of morphemes. Word phrase boundaries are marked by spaces. The underlined morpheme in the sentence (actually, "*i*" is copula) should be concatenated to the noun phrase (*NP*) "*gwijunghan geos* (valuable thing)" only after that *NP* has been syntactically combined from the adnominal "*gwijunghan*(valuable)" and the noun "*geos*(thing)." Since Korean is a head-final language, the succeeding morpheme "*i*" concatenated to the *NP* construct[3] becomes the syntactic head, and the resultant construct acts as a predicate of the sentence. This characteristic, namely post-syntactic morpheme concatenation (PSMC), arises because the agglutination of the copula occurs after the two syntactic constructs were combined together to an *NP*.

Rules adopting a word phrase as the unit of analysis encounter difficulties because of PSMC. Therefore, approaches that apply this kind of rules preprocess a source corpus and input sentences to reconfigure the word phrases so as to make them cover PSMC, which in turn causes an undesirable source distortion [1, 2]. This limitation provides an opportunity to morpheme-based rules for the analysis of Korean.

By the way, Korean morphemes are split into lexical morphemes (such as nouns or verbs), which represent the state of a construct, and grammatical morphemes (such as postpositions or verb endings) that take a syntactic role of

[1]You can see the explanation of every symbol in Appendix A.

[2]SUBJ(subjective), REL_CLS(relative clause), and DECL(declarative) denote grammatical functions.

[3]We use the term "construct" to mean a chunk, a constituent, or whatever being generated during tree construction.

the construct in a sentence[4]. Since the both kinds of morphemes serve distinctively to the construction of a sentence, we need to deal with them because of distinction in writing rules.

Restricted form of phrase structure grammar (RPSG) [3, 4] uses morpheme-based rules with restricted forms such as $A \rightarrow B+\tau$ and $A \rightarrow B+\gamma\ C$ so that it can cover PSMC. However, it provides no label for a syntactic role such as subjective or objective. It happens because RPSG cares only about lexical morphemes such as syntactic labels and treats grammatical morphemes (except for derivational affixes) as a juncture between lexical morphemes. Consequently, in parsing a sentence with RPSG, we need to see the grammatical morpheme connected to the *NP* in order to determine the *NP*'s thematic role in the sentence. In Korean, however, the lexical morpheme "***sigan***(time)" acts as a unit of scrambling only after being accompanied by the grammatical morpheme "+***i***(SUBJ)." Moreover, every noun phrase in a sentence must eventually be associated with a specific thematic role. Therefore, the subjective noun phrase "*sigan*+***i***" is required to be given a label as a construct. In addition, only eight syntactic labels in RPSG[5] simplify the rules of the language, resulting in the decrease of discriminative power of the rules.

We devise a morpheme-based rule format with an automatic labeling algorithm using POS tags to form a syntactic label of a construct. Morpheme-based rules can cover PSMC. The label comes directly from POS tags so that it can reflect surface (i.e., POS tag) information onto the construct without discarding them. Each label is of binary form like *DetNoun* → *Det Noun*, where *Det* and *Noun* come from each label of the two sub-constructs, determiner and noun. Each label is formed to represent both the state and the role of the construct in a sentence if possible. This scheme makes the resultant label keep sub-constructs' information as well as the label of the construct itself. Moreover, a construct such as "*sigan*+***i***", which wasn't given a label in the RPSG, is given a thematic role in the label such as *SubjNoun* → *Noun* + *Subj*. In addition, the proposed rule format is quite similar to that of dependency grammar (DG), which is popularly used for Korean, so that it can readily be used in a DG. The proposed rule description scheme with an automatic labeling algorithm is effective for the syntactic analysis of Korean.

---

[4]Exceptions to the grammatical morphemes are derivational affixes (e.g., "***ha***/xsm" transforms the *NP* "***gwijung***/ncps" into *ADJP* "***gwijung***+***ha***") and copula ("***i***/jp" that transforms the *NP* "***gwijunghan geos***" to *VP* "***gwijunghan geos***+***i***"), which represent the states of the transformed constructs.

[5]*S* (sentence), *NP* (noun phrase), *VP* (verbal phrase), *ADJP* (adjectival phrase), *ADVP* (adverbial phrase), *MODP* (modifier phrase), *AUXP* (auxiliary verbal phrase), and *IP* (interjectival phrase)

In the next section, we present previous approaches to parsing Korean. After we propose our rule description and labeling scheme, we compare the proposed scheme with related works. In Section 4, we provide parsing experiments on 31,080-sentence corpus to test performance, and compare the results with existing ones. Finally, we conclude the work with future research areas.

2. Previous Approaches to Rule Representation for Parsing Korean

Since Korean is an agglutinative language with partially free word order, DG [1, 2, 5] and RPSG [3, 4] have often been used for the syntactic analysis. In this section, we will review the characteristics of DG and of RPSG for the analysis of Korean, and their rule description schemes.

2.1. Representation of dependency grammars

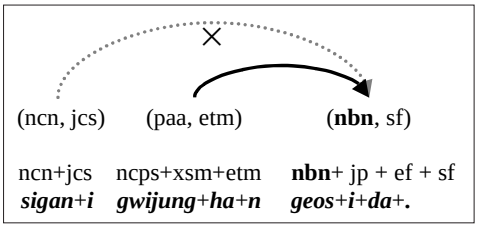
DG [6] is based on the dependency relation between two words: a head and a dependent. The rule description scheme in DG is not an important issue since the analysis uses only the relationships between those identified heads and dependents. However, because Korean is an agglutinative language where several morphemes (MPs) form a word phrase (WP) by agglutination, how to define and represent heads and dependents is still an issue to resolve. According to the unit of head and dependent, the DGs for Korean are classified into two types: MP-based [5] vs. WP-based [1, 2] DGs.

MP-based DGs cover the PSMC characteristic of Korean. However, they do not discriminate lexical morphemes that represent the state of a construct from grammatical morphemes that take a syntactic role of the construct in a sentence. In addition, they do not distinguish the dependencies invoked by agglutination and the dependencies syntactically invoked between WPs.

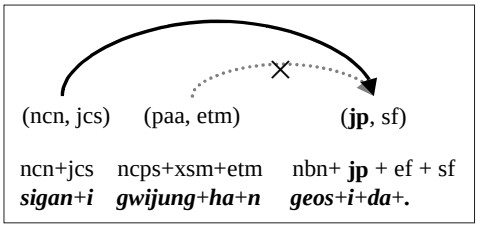
In WP tag-based DGs, the basic unit for syntactic analysis is a WP. Because a WP may modify or may be modified by other WP, each WP should keep two kinds of syntactic labels of the form (*ltag*, *rtag*). Since a head WP follows a dependent WP in Korean, *ltag* represents the WP's label as a head to a preceding WP (dependent) and *rtag* as a dependent label to a following WP (head). However, the selection of representative *ltag* and *rtag* of a WP has some problems as we can see in Figure 1.

First, the selection of representative tags is not only applied to corpus preparation but also to pre-parsing stage in order to transform every input sentence (i.e., tag sequences) into corresponding sequences of WP labels. Second, a WP including 'jp' (postposition) or 'etn' (derivational verb ending for nominalization) encounters a problem. If we select (nbn, sf) as the representative label of the

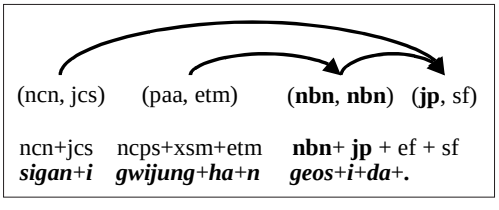
third WP in Figure 1(a), then the subjective noun phrase "***sigan***(time)+***i***" loses its head "+***i***(be)." On the other hand, if we select (jp, sf) as the label of the third WP in Figure 1(b), then the adnominal phrase "***gwijung***+***ha***+***n***(valuable)" loses its head "***geos***(thing)." To remedy this problem, WP tag-based DG splits such WP into two as shown in Figure 1(c), which in turn causes an undesirable source distortion.



(a) When the third WP's label is set to (nbn, sf)



(b) When the third WP's label is set to (jp, sf)



(c) When the third WP is split into (nbn, nbn) and (jp, sf)

Legend:

ncn:	common noun
ncps:	stative noun
jcs:	subjective postposition
jp:	predicative postposition
xsm:	adjectival suffix
etm:	modifying ending
ef:	final ending
sf:	sentential mark

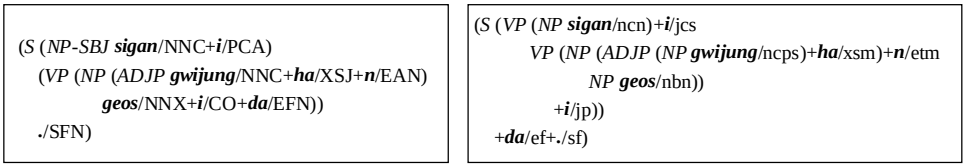
Figure 1. Possible representations of sentence (1) in WP tag-based DG.

2.2. Phrase structure grammars

Phrase structure grammars (PSGs) has no restrictions on the number of terminals and non-terminals in the right-hand side (RHS) of a rule. This entails the shortage of typicality in writing grammar rules. In addition, a statistical PSG has the preference of a smaller tree to a bigger tree due to the variance in the length of the RHS.

RPSG is proposed to cope with free word order characteristic of Korean [3, 4]. RPSG restricts grammar rules to three types: type 1 rules ($A \rightarrow B+\tau$) to describe derivation; type 2 rules ($A \rightarrow B+\gamma\ C$) to represent inter-constituent relations; and type 3 rules ($A \rightarrow A_1+\gamma\ A_2+\gamma \ldots A_n$) for coordination. Figure 2 represents sentence (1) in PSG according to Penn Korean Treebank [7] and in RPSG according to KAIST corpus [8].

Both PSG and RPSG confine syntactic labels to artificially defined small sets (8 labels in [3, 4] and 11 labels in [7]). It results in the unintentional loss



(a) PSG (b) RPSG

Figure 2. PSG and RPSG representations of sentence (1).

of intrinsic information melted in the POS tags, because the POS tags themselves reflect not only syntactic information but also morphotactic and semantic information [9].

The PSG structure does not properly describe the PSMC characteristic in the *NP* "***gwijung***+***ha***+***n***(valuable) ***geos***(thing)" while RPSG covers it using type 2 rule *NP* → *ADJP*+*etm NP*. Besides, PSG does not transform *NP* to *VP* explicitly for "[***gwijung***+***ha***+***n***(valuable) ***geos***(thing)]+***i***(be)" whereas RPSG applies type 1 rule *VP* → *NP*+*jp* for the same purpose. On the other hand, PSG represents the subjective noun phrase "***sigan***(time)+***i***" as *NP-SBJ* while RPSG provide no label for it. Since RPSG cares lexical morphemes as syntactic labels and treats grammatical morphemes only as a juncture between lexical morphemes, it cannot represent the unit of scrambling of a sentence nor it can give a thematic role therein.

3. Composite Labels for Parsing Korean

In this paper, we use the term "composite label" to point out the fact that the label *C* of a binary branching rule *C* → *L R* is composed from those of sub-contract *L* and of sub-construct *R*. In this section, we will present the forms of binary branching rules and a method of automatic generation of composite labels for the rules.

3.1. Binary rules

Based on the reviews in the previous section, we adopt binary branching rules in this paper since binary branching rules are suitable for free word-order language. In designing a rule description scheme, we take care of the following considerations to be incorporated into the scheme:

- Reflection of Korean WP structures on rules. In Korean, a grammatical morpheme cannot appear by itself but always attaches to a construct by

agglutination. Therefore, a grammatical morpheme does not need to be given an initial label, while a lexical morpheme is given a label. Rules also have to show agglutination and spacing by infix operators '+' and ' '.

- PSMC. The grammar should sublate a WP-based rule format. Each construct should be given a label. Each label of a construct should stand for both the state and the role of that construct.
- Statistical invariance in probabilistic parsing. Each candidate parse tree should comprise the same number of rules. It implies the length of RHS to be constant. It also achieves the typicality of a grammar.
- Use of POS tag information in labeling. The syntactic labels are to be designed to use POS information. This means that we do not define a label independently of POS tags.

A context-free grammar is a 4-tuple (T, N, G, S), where T and N are two finite disjoint sets of terminal (POS tag) and non-terminal (syntactic label) symbols, respectively, $S \in N$ is the start symbol, and G is a finite set of rules. In this paper, T is the KAIST tag set of 55 POS tags [9].

A rule is of binary form $C \rightarrow L\ R$ for the left-hand side (LHS) label C to represent both dependency information and its syntactic property. Basically, L is a dependent, and R is the head of the RHS. In addition, a lexical morpheme needs to be given a label while a grammatical morpheme does not. In order for a label C to represent the head of the RHS, the state of the construct (such as *NP*, *VP* in PSG), and the role of that construct (such as subject, object, etc.) at the same time, we need to devise a composite label of the form C_LC_R. Consequently, the rule $C \rightarrow L\ R$ is transformed to $C_LC_R \rightarrow L_LL_R\ R_LR_R$. With the fact that Korean has infix operators for agglutination ('+') and spacing (' '), we can define three rule types:

- Type 1 (unary rule): $C_LC_R \rightarrow f$, where $C_LC_R \in N$, $f \in T$ and f is a lexical morpheme.
- Type 2 (intra-WP rule): This type of rules covers all types of intra word-phrases. This type can be divided into three sub-types as follows:
  2a: $C_LC_R \rightarrow e + R_LR_R$;
  2b: $C_LC_R \rightarrow L_LL_R + e$; and
  2c: $C_LC_R \rightarrow L_LL_R + R_LR_R$,

  where C_LC_R, L_LL_R, $R_LR_R \in N$, $e \in T$, and e is a grammatical morpheme.
- Type 3 (inter-WP rule): $C_LC_R \rightarrow L_LL_R\ R_LR_R$, where C_LC_R, L_LL_R, $R_LR_R \in N$.

Table 1 shows examples of three types of rules, and their rules will be described in detail in the next section.

Table 1. Examples for each type of rules.

rule type	example	rule
1 ($C_LC_R \rightarrow f$)	***sigan***(time)/ncn	*0Ncn* → *ncn*
2a ($C_LC_R \rightarrow e + R_LR_R$)	***sil***(real)/xp + (*0Ncn* ***sigan***(time))	*1Ncn* → *xp* + *0Ncn*
2b ($C_LC_R \rightarrow L_LL_R + e$)	(*0Ncn* ***sigan***(time)) + ***deul***/xsn (*0Ncn* ***sigan***(time)) + ***i***(SUBJ)/jcs	*1Ncn* → *0Ncn* + xsn *jcsNcn* → *0Ncn* + *jcs*
2c ($C_LC_R \rightarrow L_LL_R + R_LR_R$)	(*0Ncpa* ***geunmu***(duty)) + (*0Ncn* ***sigan***(time))	*NcpaNcn* → *0Ncpa* + *0Ncn*
3 ($C_LC_R \rightarrow L_LL_R\,R_LR_R$)	(*etmPaa* ***gwijunghan***(valuable)) (*0Nbn* ***geos***(thing))	*EtmNbn* → *etmPaa 0Nbn*

3.2. Automatic generation of composite labels

In this subsection, we will describe how to generate a composite label for binary rules. To select proper composite labels, we need to observe the following constraints: any component labels should not be null; C_L and C_R may not take the same information to avoid duplication; and, the combination of component labels to make a composite label C_LC_R should be in a uniform way. Both C_L and C_R are notated using morphemes themselves.

We denote *L* for the left sub-construct and *R* for the right sub-construct. Subscript $_{State}$ means the state tag of the construct. Subscript $_{Role}$ means that the sub-construct having the role tag is combined to that construct as a dependent, and subscript $_{role}$ stands for the sub-construct with the role tag is agglutinated to the construct as a syntactic head.

First, we define a state and a role of a construct using POS tags. We can explicitly map which morpheme indicates a state and which one indicates a role as in Table 2. States are the same as lexical morphemes except that *Jp* is included therein. A state indicator starts with an upper case letter. Roles are in lower case letters. In other words, '*jcs*' is a role indicator, but '*Jcs*' is not.

Table 2. Indicators for state and role.

indicator	
state	*Ncpa, Ncps, Ncn, Nq, Nbu, Nbn, Npp, Npd, Nnc, Nno, F, Ii, Pvd, Pvg, Pad, Paa, Px, Mmd, Mma, Mad, Maj, Mag,* ***Jp***
role	*jcs, jco, jcc, jcm, jcv, jca, jcj, jct, jcr, jxc, jxt, jxf,ep, ecc, ecs, ecx, etn, etm, ef*

When representing a composite label of a construct, we will use lower case letters for a role, but use an upper case letter for the first character of a state. For example, when a generated composite label of a construct is '*jcsNcn*,' the state of the construct is *Ncn*, and the role is *jcs*. However, a label '*EtmNbn*' has state *Nbn* with no role because the word '*Etm*' starting with upper case letter '*E*' is not a role. Instead, this label means that a bound noun 'nbn' has been modified by an adnominal phrase '*etm**.'

Based on these notations, we can define a method for generating a composite label according to the type of rules as follows:

- Case 0 (Initial case: a sub-construct has only one state.): type 1, type 2a with a derivational prefix, and type 2b with a derivational suffix belong to this case. We use '0' for type 1 rules, and '1' for type 2 rules.

Case 0		example	state	role	LHS label
Type 1	*L*		×	×	$0R_{State}$
	R	"***sigan***/ncn"	O (*Ncn*)	×	(0*Ncn*)
	rule	*0Ncn* → *ncn*			
Type 2a	*L*	"***sil***/xp"	×	×	$1R_{State}$
	R	+(*0Ncn* "***sigan***/ncn")	O (*Ncn*)	×	(1*Ncn*)
	rule	*1Ncn* → *xp* + *0Ncn*			
Type 2b	*L*	(*0Ncn* "***sigan***/ncn")	O (*Ncn*)	×	$1L_{State}$
	R	"+***deul***/xsn"	×	×	(1*Ncn*)
	rule	*1Ncn* → *0Ncn* + *xsn*			

- Case 1 (*L* has only state and *R* has only role.): Type 2b (where *e* is not jp) belongs to this case. Composite label is $R_{role}L_{State}$, which means that the role of the resultant construct would be R_{role} and the state is L_{State}.

Case 1		example	state	role	LHS label
Type 2b	*L*	(*0Ncn* "***sigan***/ncn")	O (*Ncn*)	×	$R_{role}L_{State}$
	R	"+***i***/jcs"	×	O (*jcs*)	(*jcsNcn*)
	rule	*jcsNcn* → *0Ncn* + *jcs*			

- Case 2 (Both *L* and *R* have only states.): Type 2b (where *e* is jp) and Type 2c are in this case. Composite label $L_{State}R_{State}$ means that the resultant construct has no role, but its state is R_{State}.

Case 2		example	state	role	LHS label
Type 2b	*L*	(*EtmNbn* "***gwijung***+***ha***+***n geos***")	O (*Nbn*)	×	$L_{State}R_{State}$ (*NbnJp*)
	R	"+***i***/jp"	O (*Jp*)	×	
	rule	*NbnJp* → *EtmNbn* + *Jp*			
Type 2c	*L*	(*0Ncpa* "***geunmu***/ncpa")	O (*Ncpa*)	×	$L_{State}R_{State}$ (*NcpaNcn*)
	R	+(*0Ncn* "***sigan***/ncn")	O (*Ncn*)	×	
	rule	*NcpaNcn* → *0Ncpa* + *0Ncn*			

- Case 3 (*L* has a state and a role, *R* has only state): Type 3 is in this case. The resultant construct has no role.

Case 3		example	state	role	LHS label
Type 3	*L*	(*etmPaa* "***gwijung***+***ha***+***n***")	O (*Paa*)	O (*etm*)	$L_{Role}R_{State}$ (*EtmNbn*)
	R	(*0Nbn* "***geos***/nbn")	O (*Nbn*)	×	
	rule	*EtmNbn* → *etmPaa 0Nbn*			

- Case 4 (*L* and *R* have both states and roles): This is an exception of Type 3.

Case 4		example	state	role	LHS label
Type 3	*L*	(*jcaNbn* "***9***+***si***+***buteo***")	O (*Nbn*)	O (*jca*)	$R_{role}R_{State}$ (*jcaNbn*)
	R	(*jcaNbn* "***6***+***si***+***ggaji***")	O (*Nbn*)	O (*jca*)	
	rule	*jcaNbn* → *jcaNbn jcaNbn*			

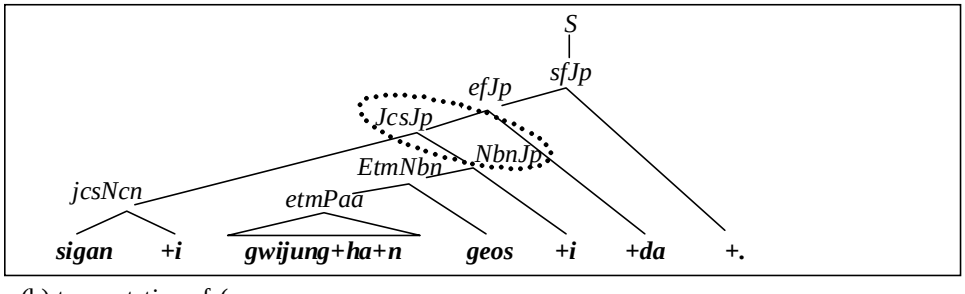
All the possible cases of combination of states and roles of a label for the binary rules will be 36 ({L_{State}, L_{Role}, L_{role}} * {R_{State}, R_{Role}, R_{role}} * {*LL*, *LR*, *RL*, *RR*}). However, there are only five valid cases of combination as in the above. It is because (i) *LL* and *RR* compositions do not exist (case 4 is the only

exception), (ii) role-to-role combinations without state information such as $L_{Role}R_{Role}$ are meaningless, (iii) a role component label always precedes a state component label in a composite label for the uniform representation, and (iv) $L_{role}R_{State}$ and $R_{Role}L_{State}$ cannot represent a construct.

The explanation of the rule *jcsNcn* → *0Ncn* + *jcs* in case 1 for the construct "***sigan***(time)/ncn+***i***(SUBJ)/jcs" is as follows. First, because "***sigan***/ncn" is a lexical morpheme, the unary rule *0Ncn* → *ncn* is applied to it. When it is combined to the following 'jcs,' the noun acts as a syntactic dependent. Since 'jcs' takes the role (SUBJ) of a construct, the composite label *jcsNcn* is formed to represent a subjective noun phrase. The lower case "*jcs*" in *jcsNcn* means that the subjective postposition follows the noun in word order. The detailed labeling algorithm for the KAIST POS tag set is given in Appendix A. Figure 3 shows the bracketed representation of sentence (1) using these composite labels.

(*S* (*efJp* (*JcsJp* (*jcsNcn* (*0Ncn* ***sigan***/ncn) + ***i***/jcs)
(*NbnJp* (*EtmNbn* (*etmPaa* (*1Paa* (*0Ncps* ***gwijung***/ncps) + ***ha***/xsm) + ***n***/etm)
(*0Nbn* ***geos***/nbn)) + ***i***/jp)) + ***da***/ef))

(a) bracketed representation of sentence (1



(b) tree notation of (a

Figure 3. Representation of sentence (1) in the proposed rule description scheme.

In Figure 3, the rule *JcsJp* → *jcsNcn NbnJp* means that the subjective nominal construct *jcsNcn* is the dependent of a predicative head *NbnJp* so that the resultant construct *JcsJp* is the predicate with a subject.

To summarize, the proposed grammar fulfills the considerations given at the start of this section. In addition, because it does not use any information other than the innate POS tag information of constructs, the grammar description scheme is open to other languages as well. One advantage of using POS tags comes from the fact that POS classification is based not only on syntactic but also on lexical and semantic information.

4. Experiments

KAIST tree-tagged corpus is made up of 31,080 sentences with 55 POS tags and 8 syntactic tags in RPSG format [3, 8, 9]. Its average number of word phrases and morphemes per sentence are 11.35 and 25.62, respectively. We transform the corpus into that of proposed rule format and divide it into 10 parts for context-free cross validation (CV). We run the MSLR parser [10] with Hancom Linux 2.2 on PC. The input to the parser is the sequence of POS tags. For evaluation, we adopt EVALB program [11] that applies PARSEVAL measure [12]. Table 3 depicts the test results of grammars on the KAIST corpus.

Table 3. Test results of the grammars on the KAIST corpus.

Grammar	test set	# labels	# rules	# ambiguity	LP	LR	Clustered LP&LR
PSG[1)]	1,000	8	5,262	8.71 e8	74.66	72.18	
RPSG-1[1)]	1,000	8	2,654	5.00 e8	74.09	72.36	
RPSG-2[2)]	31,080 CV	8	2,381	3.76 e14	75.39	72.83	
MP-based DG[3)]	31,080 CV	55	1,027	7.97 e7	75.32	75.32	
Proposed[4)]	31,080 CV	726	10,735	4.69 e6	75.58	75.58	79.30

Note:

(1) Tested with a chart parser on UltraSparc1 and reported in [3].

(2) Cross validation result of RPSG in the same environment as in (1).

(3) Tested with MSLR parser on PC. Its performance metrics are unlabeled precision and unlabeled recall.

(4) Tested with MSLR parser on PC. Clustered LP&LR are explained later in this section.

The results of the first two experiments come from [3]. The third experiment (RPSG-2) is run in the same environment as the second experiment for cross validation. PSG and RPSG use 8 syntactic labels. Their results are around 75% accuracy for Korean, which conform to the presupposition in English such as probabilistic context-free parsing for English yields around 75% accuracy for precision and recall [13]. (For example, the labeled precision (LP) and the labeled recall (LR) for *The Wall Street Journal* section 23 of the Penn Treebank are 74.3% and 70.1%, respectively [14].)

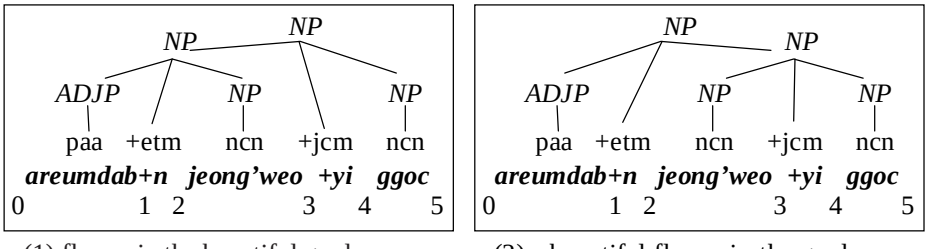
The remaining experiments have been done using MSLR parser. MP tag-based DG has 55 labels because each construct is represented by the head's POS tag. Its unlabeled precision and recall are slightly better than 75%.

The last experiment has been run using the proposed rule description scheme. The number of automatically extracted labels of the proposed grammar is 726.

As we can denote a label by C_LC_R, and C_R always represents a state, there can be 31 candidate tags for C_R (symbol, nominal, foreign, interjection, predicate, modifier, and copula of the Table A.1). 68 candidate tags for C_L can also be estimated in the same way. Among the possible 2,108 labels, there are morphotactic and syntactic connectivity restrictions in combining morphemes or constructs so that only 726 labels are found in the corpus. For example, *jcsPaa* is an erroneous label because it means the subjective postposition '*jcs*' follows the stem of an adjective 'paa.' But, such erroneous labels are still included in the grammar for the experiments if they were extracted from the corpus.

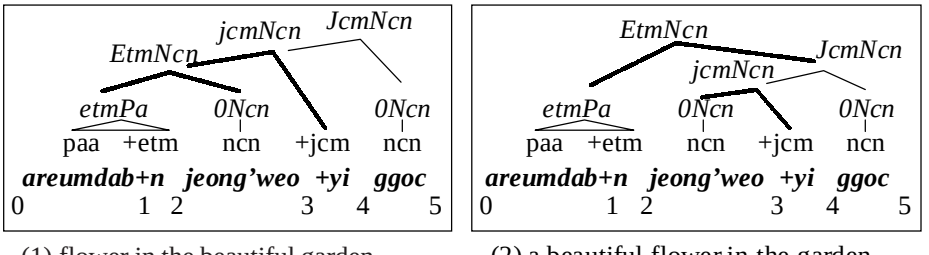
By the way, the estimation of labeled precision and labeled recall depends on the classification of non-terminals. For example, Figure 4 depicts ambiguous parses of the sentence (2) in RPSG and in the proposed rule scheme.

(2) (a) ***areumdab***/paa+***n***/etm ***jeong'weon***/ncn+***yi***/jcm ***ggoc***/ncn
(b) beautiful+MOD[6] garden+POS flower
(c) a flower in the beautiful garden; a beautiful flower in the garden



(1) flower in the beautiful garden (2) a beautiful flower in the garden

(a) Two possible structures in RPSG



(1) flower in the beautiful garden (2) a beautiful flower in the garden

(b) Two possible structures in the proposed rule scheme

Figure 4. Tree structures for sentence (2).

[6]MOD (modifier), POS (possessive) are grammatical functions.

In PSG and RPSG, as depicted in Figure 4(a), two possible parses use the same rules: *NP* → *ADJP+etm NP* and *NP* → *NP+jcm NP*. However, if we scrutinize the rules, those rules in the figure are different from each other. Figure 4(a1) has *NP*[0,3] → *ADJP*[0,1] + *etm NP*[2,3] and *NP*[0,5] → *NP*[0,3] + *jcm NP*[4,5], whereas Figure 4(a2) has *NP*[2,5] → *NP*[2,3] + *jcm NP*[4,5] and *NP*[0,5] → *ADJP*[0,1] + *etm NP*[2,5]. If we assume that the parse in Figure 4(a1) is the gold parse and the other in Figure 4(a2) is the output parse, then both the two rules in the output parse are incorrect rules. However, LP and LR according to [12] say that only *NP*[2,5] → *NP*[2,3] + *jcm NP*[4,5] is an incorrect rule because its span does not match that of the same rule in the gold parse, and that *NP*[0,5] → *ADJP*[0,1] + *etm NP*[2,5] is correct because both the label "*NP*" and the span "[0,5]" match *NP*[0,5] → *NP*[0,3] + *jcm NP*[4,5] in the gold parse, even though the two rules are quite different.

In Figure 4(b1) and 4(b2), we can see that the three rules in the output parse (*jcmNcn*[2,4], *JcmNcn*[2,5], and *EtmNcn*[0,5]) are incorrect because the labels and the spans do not match those in the gold parse. *EtmNcn*[0,5] in the output, for example, does not match *JcmNcn*[0,5] in the gold parse. It happens because the proposed rule description scheme propagates the sub-construct's information onto the label of the construct so that the error in the lower construct gives an impact to the estimation of LP and LR in the higher construct. Even with this somewhat different basis of estimation, the LP and LR of the proposed scheme get 75.58% accuracy, which is still higher than those of PSG and RPSG.

Since LP and LR are estimated on the grammar's non-terminals and RPSG uses only 8 non-terminals, LP and LR of RPSG are estimated upon the 8 corresponding labels. Based on this idea, we can define clustered LP and clustered LR. In other words, all the composite non-terminals such as *EtmNcn* in the proposed scheme can be clustered to corresponding non-composite non-terminals such as *Ncn*. Then, 726 composite non-terminals shrink to 55 non-composite ones. In this clustered method, both *EtmNcn*[0,5] in the output parse and *JcmNcn*[0,5] in the gold parse shrink to the same *Ncn*[0,5]. The clustered accuracy for the 75.58% accuracy in Table 3 is 79.30%. The results can be even higher if we transform those 726 non-terminals into 8 correspondents. Furthermore, because DG only needs to count correct brackets and the proposed scheme can be considered as a kind of DG representation scheme, we can get 79.52% accuracy from that point of view.

5. Discussion and Future Research

We analyzed Korean syntactic characteristics and existing grammars. The proposed rule format and labeling adopt binary branching rules with composite labels

automatically generated from POS tags. The resultant grammar naturally represents both the state and the role of a construct, and both the dependency and the syntactic information of that construct during the parsing. It also meets the PSMC characteristic and avoids a statistical bias.

The scheme uses only the POS information in the input, and shows 75.58% LP and LR, and 79.30% clustered accuracy, which are above 5% higher results than that of RPSG and DG.

The rule description scheme does not use contextual information such as head-to-head collocation, outside context, distance information, or argument structures at all. Those information can readily be melted in the scheme to yield higher results. In addition, because it does not use any information other than the innate POS tag information of constructs, the rule description scheme is open to other languages as well.

By the way, there is still more to do to enhance the parsing results. For the Korean corpus, some words must be given a new tag. Splitting '*pvg*' (verb) into transitive and intransitive verbs is one thing, since otherwise the identification of its syntactic usage would be left to the parser's burden. Interim tags will also be useful: *NOM* for nominalization and *CLS* for clausification. Sentences (3) and (4) are such examples.

(3) (a) (*jcsNOM* (*jcaNpp* ***Cheolsu***/npp+***ggaji***/jca)+***ga***/jcs) ***habgyeog***/ncpa+***i***/jp+***da***/ef
(b) ***Cheolsu***+downto+SUBJ success+be+DECL
(c) Downto ***Cheolsu*** are accepted.

(4) (a) (*jcsCLS* (*efPvg* (*efPvg* ***jug***/pvg+***neunya***/ef) (*efPvg* ***sal***/pvg+***neunya***/ef))+***ga***/jcs) ***munje***/ncn+***i***/jp+***da***/ef
(b) die+QUEST[7] live+QUEST+SUBJ problem+be+DECL
(c) To be or not to be is the problem.

Our future research areas are two-fold. First, because the proposed rule description scheme uses a generic labeling algorithm and its composite labels use only the POS tags of the target language, it can be applied not only to free word order languages including Japanese but also to configurational languages such as English. We expect that the experiment on the Penn Treebank with the proposed scheme will also get better parsing results.

Morpho-syntactic analysis is another area to tackle. MSLR parser is an integrated morphological and syntactic analysis of Japanese using GLR algorithm.

[7]QUEST (interrogative) is a grammatical function.

They use a connectivity matrix between POS tags, which is linguistically prescribed. Our proposed scheme is POS tag-based and binary, and it uses the same tag set for both morphological and syntactic labels. In addition, the training of syntactic rules naturally includes the morphotactic connectivity probability distribution. Besides, binary rules make it possible to integrate n-gram tagger into the process. Therefore, integrating morphological analysis, tagging, and syntactic analysis is not a difficult problem.

Acknowledgements

This research was supported by the Korea Ministry of Culture and Atheletics.

Appendix A. Composite Category Labeling Algorithm for KAIST POS Tag Set

This appendix describes the labeling algorithm in detail for the proposed scheme using KAIST POS tag set in Table A.1. The initial of each subset in the parentheses is used for the labeling algorithm.

lexical morphemes	(S)ymbol	sp(*pause*), sf(*full stop*), sl(*left quotation & parenthesis mark*), sr(*right quotation & parenthesis mark*), sd(*dash*), se(*ellipsis*), su(*unit*), sy(*other symbols*)
	(N)ominal	ncpa(*active-predicative common noun*), ncps(*stative-predicative common noun*), ncn(*non-predicative common noun*), nq(*proper noun*), nbu(*unit bound noun*), nbn(*non-unit bound noun*), npp(*personal pronoun*), npd(*demonstrative pronoun*), nnc(*cardinal numerals*), nno(*ordinal numerals*)
	(F)oreign	f(*foreign word*)
	(I)nterjection	ii(*interjection*)
	(P)redicate	pvd(*demonstrative verb*), pvg(*general verb*), pad(*demonstrative adjective*), paa(*attributive adjective*), px(*auxiliary verb*)
	(M)odifier	mmd(*demonstrative adnoun*), mma(*attributive adnoun*), mad(*demonstrative adverb*), maj(*conjunctive adverb*), mag(*general adverb*)
grammatical morphemes	(J)osa (postposition)	jcs(*subjective case*), jco(*objective case*), jcc(*complemental case*), jcm(*adnominal case*), jcv(*vocative case*), jca(*adverbial case*), jcj(*conjunctive*), jct(*comitative*), jcr(*quotative*), jxc(*common auxiliary*), jxt(*topical auxiliary*), jxf(*final auxiliary*), jp(*predicative case*)
	(E)nding	ep(*prefinal*), ecc(*coordinate conjunctive*), ecs(*subordinate conjunctive*), ecx(*auxiliary conjunctive*), etn(*nominalizing*), etm(*adnominalizing*), ef(*final*)
	(A)ffix	xp(*prefix*), xsn(*noun-derivational*), xsv(*verb-derivational*), xsm *adjective-derivational*), xsa(*adverb-derivational*)

Since each non-terminal has a composite label, we can rewrite three types of rules in the main text as follows:

- Type 1 (unary rule): $C_LC_R \rightarrow f$, where $C_LC_R \in N$, $f \in T$ and f is a lexical morpheme. (The sentential rule $S \rightarrow C_LC_R$ belongs to this type with S, $C_LC_R \in N$.)
- Type 2 (intra-WP rule): $C_LC_R \rightarrow L_LL_R + R_LR_R$ (type 2a), $C_LC_R \rightarrow e + R_LR_R$ (type 2b), or $C_LC_R \rightarrow L_LL_R + e$ (type 2c), where C_LC_R, L_LL_R, $R_LR_R \in N$, $e \in T$ and e is a grammatical morpheme.
- Type 3 (inter-WP rule): $C_LC_R \rightarrow L_LL_R\ R_LR_R$, where C_LC_R, L_LL_R, $R_LR_R \in N$.

In Korean, we can always identify the dependency relation automatically between the two sub-trees if their POS's are given [3]. So, we can derive a generic labeling algorithm from the POS tag set.

Let us define some notations first. A function $C_R = Cat(f)$ returns an implicit category C_R for a tag f. $LB(C_i)$ returns the left-branching value of the category C_i, and $RB(C_i)$ returns the right-branching value of C_i. $HC(C_LC_R)$ returns the head-child of the C_LC_R, i.e., returns L_LL_R or R_LR_R which contains C_R.

A.1. Labeling C_L and C_R

```
switch (RHS type)
{
  case 1 : // unary rule
      if (end of sentence) {C_L = null; C_R = Sentence; break;}
      else {C_L = '0'; C_R = Cat(f); break;}                          (Eg. 1)
  case 2b : // derivation
      if (R_R ∈ {N, F} && e == xp) {C_L = '1'; C_R = R_R; break;}     (Eg. 2)
  case 2c : // derivation
      if (L_R ∈ {N, F} && e == xsn) {C_L = '1'; C_R = Ncn; break;}    (Eg. 3)
      elseif (L_R ∈ {N, F} && e == xsv) {C_L = '1'; C_R = Pvg; break;}(Eg. 4)
      elseif (L_R ∈ {N, F} && e == xsm) {C_L = '1'; C_R = Paa; break;}(Eg. 5)
      elseif (L_R ∈ {N, F} && e == xsa) {C_L = '1'; C_R = Mag; break;}(Eg. 6)
      elseif (L_L == etn && L_R ∈ {P} && e ∈ {J - jp})
          {C_L = LB(Cat(e)); C_R = Ncn; break;}                       (Eg. 7)
      elseif (L_L == etn && L_R ∈ {P} && e == jp)
              {C_L = Ncn; C_R = Jp; break;}                           (Eg. 8)
  case 3 : // derivation
      if (R_L == etn && R_R ∈ {P}) {C_L = L_L; C_R = Ncn; break;}     (Eg. 9)
      elseif (L_L == etn && L_R ∈ {P}) {C_L = Ncn; C_R = R_R; break;} (Eg. 10)
```

default : // otherwise
switch (cell value of Table A.2) // labeling C_L
{
case L :
if ($L_L \in \{0, 1, N, F, I, M\} \parallel L_L \in LB(Cat(\{J, E\}))$)
$\{C_L = LB(Cat(L_R));\}$ (Eg. 11)
elseif ($L_L \in \{S\}$)
if ($L_R \in \{P, Jp\}$)
$\{X_L X_R = HC(L_L L_R);$
while ($X_L \in \{S\}$) $X_L X_R = HC(X_L X_R);$
$C_L = LB(X_L);\}$ (Eg. 12)
else $\{C_L = LB(Cat(L_R));\}$ (Eg. 13)
else // $L_L \in \{J, E\}$
if (L_L == jxc && $L_R \in \{M\}$) $\{C_L = LB(L_R);\}$ (Eg. 14)
else $\{C_L = LB(L_L);\}$ (Eg. 15)
break;
case R :
if ($R_R \in \{J, E\}$)
if (R_R == jxc && $L_L \in RB(Cat(\{J, E\}))$)
$\{C_L = RB(L_L);\}$ (Eg. 16)
else $\{C_L = RB(Cat(R_R));\}$ (Eg. 17)
else $\{C_L = RB(Cat(R_R));\}$ (Eg. 18)
}
// labeling C_R
choose the value in the corresponding cell of Table A.3 as C_R; (Eg. 19)
}

Table A.2. Selecting L ($L_L L_R$) or R ($R_L R_R$) for labeling C_L.

L \ R	S	N	F	I	P	M	Jp	J	E	A
S	L	L	L	L	L	L	L	R	R	R
N	R	L	L	L	L	-	L	R	R	-
F	R	L	L	L	L	-	L	R	R	-
I	R	L	L	L	L	-	L	R	R	-
P	R	L	L	L	L	-	L	R	R	-
M	R	L	L	L	L	L	L	R	R	-
Jp	R	L	L	L	L	-	L	R	R	-
J	-	-	-	-	-	-	-	-	-	-
E	-	-	-	-	-	-	-	-	-	-
A	-	-	-	-	-	-	-	-	-	-

Table A.3. Selecting C_R from L_R or R_R

L *R*	*S*	*N*	*F*	*I*	*P*	*M*	*Jp*	*J*	*E*	*A*
S	*R*	*R*	*R*	*R*	*R*	*R*	*R*	*L*	*L*	-
N	*L*	*R*	*R*	*R*	*R*	-	*R*	*L*	*L*	-
F	*L*	*R*	*R*	*R*	*R*	-	*R*	*L*	*L*	-
I	*L*	*R*	*R*	*R*	*R*	-	*R*	*L*	*L*	-
P	*L*	*R*	*R*	*R*	*R*	-	*R*	*L*	*L*	-
M	*L*	*R*	*R*	*R*	*R*	*R*	*R*	*L*	*L*	-
Jp	*L*	*R*	*R*	*R*	*R*	-	*R*	*L*	*L*	-
J	-	-	-	-	-	-	-	R	-	-
E	-	-	-	-	-	-	-	R	-	-
A	*R*	*R*	*R*	*R*	*R*	*R*	-	-	-	-

A.2. Labeling examples

(Eg. 1) *0Ncn* [time] → *ncn* [***sigan***(time)]
(Eg. 2) *1Ncn* [new technology] → *xp* [***sin***(new)] + *0Ncn* [***gisul***(technology)]
(Eg. 3) *1Ncn* [times] → *0Ncn* [***sigan***(time)] + *xsn* [***deul***(plural)]
(Eg. 4) *1Pvg* [think] → *0Ncpa* [***saenggag***(thought)] + *xsv* [***ha***(do)]
(Eg. 5) *1Paa* [valuable] → *0Ncps* [***gwijung***(value)] + *xsm* [***ha***(be)]
(Eg. 6) *1Mag* [eternally] → *0Ncps* [***yeong'weon***(eternity)] + *xsa* [***hi***(adverbial)]
(Eg. 7) *jcoNcn* [coming] → *etnPvg* [***o***+***gi***(coming)] + *jco* [***reul***(OBJ)]
(Eg. 8) *NcnJp* [is to come] → *etnPvg* [***o***+***gi***(coming)] + *jp* [***i***(copula)]
(Eg. 9) *JcjNcn* [abdominal dropsy or swell] → *jcjNcn* [***bogsu***+***na***(abdominal dropsy or)] *etnPvg* [***bus***+***gi***(swell)]
(Eg. 10) *NcnNbn* [because of eating] → *etnPvg* [***meog***+***gi***(eating)] *0Nbn* [***ddaemun***(reason)]
(Eg. 11) *NbnJp* [be valuable thing] → *EtmNbn* [***gwijung***+***ha***+***n geos***(valuable thing)] + *jp* [***i***(copula)]
(Eg. 12) *EccPaa* [smart and diligent] → *spJp* (*eccPaa* [***joh***+***go***(smart and)] + *0Sp* [,]) *0Paa* [***bujireonha***(diligent)]
(Eg. 13) *NcpaNcn* [politics, economy] → *spNcpa* (*0Ncpa* [***jeongci***(politics)] + *0Sp* [,]) *0Ncn* [***gyeongje***(economy)]
(Eg. 14) *MagPaa* [too long] → *jxcMag* [***neomu***+***na***(too)] *0Paa* [***gil***(long)]
(Eg. 15) *JxtJp* [he is a student] → *jxtNpp* [***geu***+***neun***(he+SUBJ)] *NcnJp* [***hagsaeng***+***i***(is a student)]
(Eg. 16) *eccJp* [being dynamic] → *eccJp* [***yeogdongjeog***+***i***+***myeonseo***(being dynamic)] + *jxc* [***do***(and)]
(Eg. 17) *jcsNcn* [time] → *0Ncn* [***sigan***(time)] + *jcs* [***i***(SUBJ)]

(Eg. 18) *spNcpa* [politics,] → *0Ncpa* [***jeongci***(politics)] + *0Sp* [,]
(Eg. 19) Omitted (Examples 11 through 18 belong to this).

References

[1] C.H. Kim, J.H. Kim, J.Y. Seo and G.C. Kim, "A Right-to-left Chart Parsing with Headable Paths for Korean Dependency Grammar", *Computer Processing of Chinese and Oriental Languages* 8 (Supplement), 1994, 105–118.

[2] K.J. Seo, K.C. Nam and K.S. Choi, "A Probabilistic Model for Dependency Parsing Considering Ascending Dependencies", *Literary and Linguistic Computing*, 13(2), 1998, 59–63.

[3] K.J. Lee, J.H. Kim and G.C. Kim, "An Efficient Parsing of Korean Sentences Using Restricted Phrase Structure Grammar", *Int. J. Computer Processing of Oriental Languages*, 11(1), 1997, 49–62.

[4] K.J. Lee, *Probabilistic Parsing of Korean Based on Language-Specific Properties*, Ph.D. Thesis, Department of Computer Science, Korea Advanced Institute of Science and Technology, Taejon, 1998.

[5] S.M. Lee and K.S. Choi, "A Reestimation Algorithm for Probabilistic Dependency Grammars", *Natural Language Engineering*, 5(3), 1999, 251–270.

[6] I.A. Melčuk, *Dependency Syntax: Theory and Practice*, State University of New York Press, 1988.

[7] C.H. Han, N.R. Han and E.S. Ko, *Bracketing Guidelines for Penn Korean TreeBank*, IRCS Report 01–10, University of Pennsylvania, 2001.

[8] *The KAIST corpus 1996-1997*, Korea Advanced Institute of Science and Tehcnology, 1997, http://korterm.org/.

[9] K.S. Choi, Y.J. Nam, J.G. Kim, Y.G. Han, S.M. Park, J.S. Kim, C.T. Lee, D.B. Kim, J.H. Kim and B.J. Choi, "A Study of the Morphological and Syntactic Tag Standard for Korean Language Information Bases", *Korean Journal of Cognitive Science*, 7(4), 1996, 43–61.

[10] H. Tanaka, T. Tokunaga and M. Aizawa, "Integration of Morphological and Syntactic Analysis Based on LR Parsing Algorithm", *Journal of Natural Language Processing*, 2(2), 1995, 59–74.

[11] S. Sekine and M. Collins, Evalb, 1997, *ftp://cs.nyu.edu/*.

[12] E. Black, S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini and T. Strzalkowski, "A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars", *Proceedings of*

Speech and Natural Language Workshop, DARPA, Pacific Grove, 1991, pp. 306–311.

[13] C.M. White, *Rapid Grammar Development and Parsing Constraint Dependency Grammars with Abstract Role Values*, Ph.D. Thesis, Purdue University, 2000.

[14] E. Charniak, "Statistical Parsing with a Context-Free Grammar and Word Statistics", AAAI, 1997, 598–603.