Developing a Korean Resource Grammar and Implementing It into the Linguistic Knowledge Building System*

Jong-Bok Kim (Kyung Hee University)
Jaehyung Yang (Kangnam University)

Jong-Bok Kim and Jaehyung Yang. 2006. Developing a Korean Resource Grammar and Implementing It into the Linguistic Knowledge Building System. ***The Journal of Linguistic Science*** **37, 23-67.** This paper reports our on-going project aiming to develop a Korean resource grammar that can provide us with the possibility of deep processing for Korean. Though there exist several reliable morphological analysers developed for Korean, not many attempts have been made to build its syntactic or semantic parser(s), partly because of its structural complexity and partly because of few reliable grammar platforms. This paper presents some results of developing a computationally feasible resource grammar for Korean, named Korean Phrase Structure Grammar (KPSG), whose basic framework is couched upon a typed feature structure grammar, HPSG (Head-driven Phrase Structure Grammar). The paper also discusses results of implementing the grammar into the LKB (Linguistic Knowledge Building) grammar platform.

Key words: Korean resource grammar, Korean Phrase Structure Grammar, computational implementation, linguistic knowledge building system

---

* Since this is a report of our on-going project of Korean grammar writing aiming for deep processing, we admit that it sometimes lacks in-depth discussion to cover more wide range of topics within the limited space. We thank the reviewers of this journal whose critical comments helped us improve the paper. All errors are of course our own. This work was supported by the Korea Research Foundation Grant funded by the Korean Government (KRF-2005-042-A00056).

1. Background

The advent of the information era in this century has escalated the importance of processing linguistic information more precisely and correctly. Recent developments in artificial intelligence, information science, and other high technology activities have made it possible to build feasible computational applications for language processing and understanding. Such applications (e.g. message extraction systems, web-based search engines, machine translation, and dialogue understanding systems) demand increasing accuracy and robustness of the grammar (or parsers) combined with sophisticated statistical processing methods.

When considering the reality that the basic units in understanding language are sentences, we could not miss the fact that building a reliable syntactic and semantic parser is a prerequisite for language processing. Although there have been several successful morphological analyzers developed for Korean, there exist few syntactic or semantic parser(s) that we can rely on. As observed by Kang (1999), the research on syntactic and semantic processing for Korean is at the beginning stage and at least 10 to 15 years behind compared to the one for English:

(1)

	English	Korean
Morphological Analyzers	Application Stage	Application Stage
Corpus	Application Stage	Research and Application Stage
Syntax/Semantic Parser	Research and Application Stage	Research Stage

As represented in the table, the research for the development of English syntactic and semantic parsers has reached a significant level that can even allow real-time applications. For example, in past projects, the ERG (English Resource Grammar), a part of the LinGO project at CSLI (Center for the Study of Language and Information), was used in the Verbmobil machine translation system and in an NSF-funded project on computer-aided speech generation for people who cannot speak because of disability (cf. Copestake and Flickinger 2000, Oepen et al. 2002).

Our on-going project, starting to fill such a gap, aims to build a general purpose system for processing the Korean language that can support both research and practical applications. Its goals also include building a broad-coverage Korean grammar that can be used both to extract precise meanings from text input and to generate well-formed text output. To achieve this, our project has so far focused on developing a computationally feasible Korean resource grammar, named KPSG (Korean Phrase Structure Grammar) here. The project has then implemented the grammar into the LKB (Linguistic Knowledge Building) system (cf. Copestake 2002) to check its computational feasibility and robustness. In what follows, we provide the basic of Korean grammar and some results of its implementation.

2. Korean Phrase Structure Grammar

2.1 General Aims

The KPSG developed in this project is a computational grammar for Korean currently under development and still in progress. As the grammatical framework, the KPSG adopts the constraint-based

grammar, HPSG (Pollard and Sag 1994, Sag, Wasow, and Bender 2003). HPSG is a constraint-based, lexicalist approach to grammatical theory that seeks to model human languages as systems of constraints on typed feature structures. In particular, the grammar adopts the mechanism of type hierarchy in which every linguistic sign is typed with appropriate constraints and hierarchically organized. The characteristic of such typed feature structure formalisms facilitates the extension of grammar in a systematic and efficient way, resulting in linguistically precise and theoretically motivated descriptions of languages in question.[1] The grammar HPSG is thus well suited to the task of multilingual development of broad coverage grammars (see Copestake and Flickinger 2000, Oopen et al. 2002).

The KPSG, couched upon such a type feature structure grammar, aims to have a broad lexical and syntactic coverage. In addition, the grammar adopts a flat semantic formalism Minimal Recursion Semantics (MRS) in representing semantics (Copestake et al. 2001). MRS offers an interface between syntax and semantics using feature structures. The formalism has syntactically flat structures and offers at the same time the possibility of the handling of scope relations.[2]

The basic tool for writing, testing and processing the KPSG is the LKB (Linguistic Knowledge Building) system (Copestake 2002). The LKB system is a grammar and lexicon development environment for use with constraint-based linguistic formalisms such as HPSG.

---

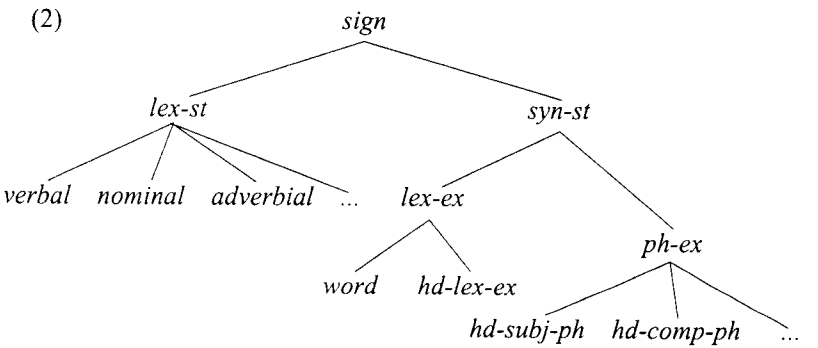
[1] We use the term 'precision' related to computational implementations and mathematical tractability. When the grammar we develop cannot be formalized, we believe it is not precise enough.

[2] The semantics is being developed in close cooperation with the LinGO English Resource Grammar. See Copestake and Flickinger 2000.

2.2 Basic Picture of the Grammar

The main components of the KPSG include grammar rules, inflection rules, lexical rules, type definitions, and lexicon.[3] As in HPSG (Sag et al. 2003), the grammar adopts the mechanism of type hierarchy in which every linguistic sign is typed with appropriate constraints and hierarchically organized. All the linguistic information is thus represented in terms of *sign*. The type *sign* is classified into subtypes as represented in a simplified hierarchy in (2):

(2)



The elements in *lex-st* type, forming the basic components of the lexicon, are built from lexical processes such as lexical rules and type definitions. Parts of these elements will be realized as *word* to function as syntactic elements. Phrases projected from *word* form basic Korean well-formed phrases including *hd-lex-ex* and the subtypes of *ph-ex* (see section 2.4). In what follows, we briefly discuss the subtypes of *lex-st*

[3] The space does not allow us to explicate the morphological system of the KPSG. As for morphology, we integrated MACH (Morphological Analyzer for Contemporary Hangul) developed by Shim and Yang (2002). This system segments words into sequences of morphemes with POS tags and morphological information. In the KPSG, it is in fact the type definitions on each morphological elements that play more crucial roles in forming morphological elements and incurring relevant grammatical information.

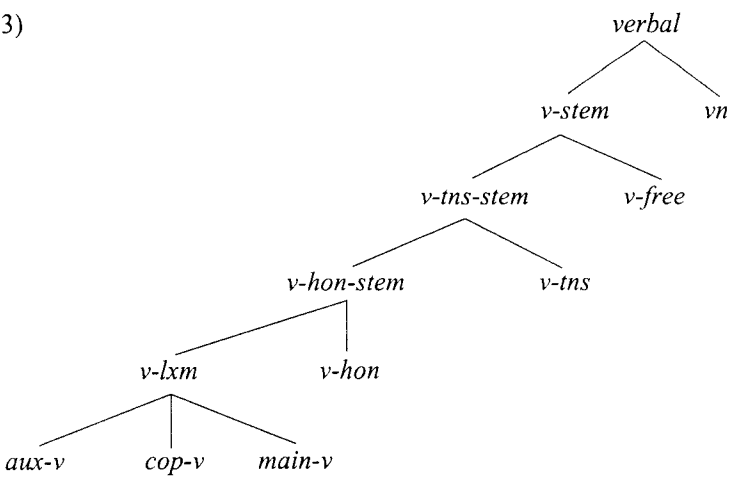
and *syn-st* and constraint specifications on types, in order to see the big picture of how the system works.

2.3 Lexicon

2.3.1 Verbal Morphology

The starting point of structuring the lexicon in the KPSG is parts of speech in the language. Like the traditional literature, the KPSG assumes *verbal*, *nominal*, *adverbial*, and *adnominal* as the subtypes of *lex-st*. These are further subclassified according to morphological properties. For example, the type *verbal* is taken to have the hierarchy given in (3):

(3)



Each of this subtype has its own reason of existence in verbal inflectional processes (cf. Kim 1998). Some of the examples for each type are given in (4):[4]

[4] The abbreviations for features and glosses adopted in this paper are follows: ACC (Accusative), ARG-ST (argument-structure), ARG (argument), COMP

(4) a. *v-lxm*: ilk- 'read'
b. *v-hon*: ilk-usi 'ilk-HON'
c. *v-pst*: ilk-usi-ess 'ilk-HON-PST'
d. *v-free*: ilk-usi-ess-ta 'ilk-HON-PST-DECL'

Such a classification aims to capture the basic verbal morphology of Korean. The verb in Korean cannot be an independent word without inflectional suffixes. The suffixes cannot be attached arbitrarily to a stem or word, but need to observe a regular fixed order. Reflecting this, the verbal morphology has traditionally been assumed to be templatic as given in (5) (see Cho and Sells 1995):

(5) V-base + (Passive/Causative) + (HON) + (TNS) + MOOD + (COMP)

As can be seen from the above template, verb suffixes mark honorific, tense, and mood functions. Morphologically, the inflectional suffixes preceding Mood are optional, but a Mood suffix obligatorily needs to be attached to a verb stem in simple independent sentences. The verbal stem and the mood suffix are thus mutually bound in the sense that the stem cannot be used uninflected in any syntactic context and that the stem should be inflected at least with a mood suffix, as seen in (6).

(6) a. ilk-(ess)-ta 'read-(PST)-DECL'
b. *ilk-ess 'read-PST'

Within the type hierarchy system in (3), the grammar place type

---

(Complementizer), CONJ (Conjunction), DECL (Declarative), DEL (Delimiter), GEN (Genitive), HON (Honorific), IMPER (Imperative), LOC (Locative), NOM (Nominative), NMLZ (Nominalizer), ORTH (orthography) PL (Plural), PST (Pst), POSTP (Postposition), PROP (Propostive), RELN (relation), SYN (syntax), SEM (semantics), SUG (Suggestive), TNS (tense).

constraints on these morphological types, some of which are given in (7):[5]

(7)

a. $v-hon \rightarrow \begin{bmatrix} \text{STEM} \begin{bmatrix} v-lxm \\ \text{ORTH } \boxed{1} \end{bmatrix} \\ \text{SYN.HEAD.HON} + \end{bmatrix}$

b. $v-tns \rightarrow \begin{bmatrix} \text{STEM} \begin{bmatrix} v-hon-stem \\ \text{SYN } \boxed{1} \\ \text{SEM.RELS } \boxed{A} \end{bmatrix} \\ \text{SYN } \boxed{1} \\ \text{SEM.RELS } \boxed{A} \oplus \boxed{B} \end{bmatrix}$

c. $v-hon \rightarrow \begin{bmatrix} \text{STEM } v-tns-stem \\ \text{SYN.HEAD.IC } bool \end{bmatrix}$

The constraints in (7a) mean that the type *v-hon* will take *v-lxm* as its stem; (7b) means that the type *v-tns* will take an instance of *v-hon-stem* as its stem. One thing to note here is that any subtypes of *v-hon-stem* can serve as the stem of *v-tns* in accordance with the type hierarchy system. The grammar also allows the instances of *v-free* to serve as an input to syntax: that is, other than *v-free*, nothing can be projected into a *word* level element that can be a syntactic element. This will make sure not fully inflected elements such as *v-hon* or *v-tns* do not appear in syntax.

These constraints eventually restrict the possible word internal structures in Korean word formation. The system could provide a clean account for the ill-formed combinations without employing mechanisms such as templates. Observe the following:

[5] The variable [B] here means tense information. The implemented feature descriptions in the LKB system are slightly different from those represented here.

(8) a. *$_{v\text{-}hon}$[$_{v\text{-}pst}$[cap-ass]-si]-ta 'catch-PST-HON-DECL'
b. *$_{v\text{-}free}$[_{v-hon}$[cap-usi]-ta]-ess 'catch-HON-DECL-PST'
c. *$_{v\text{-}hon\text{-}stem}$[$_{v\text{-}hon}$[cap-usi]-usi]-ess]-ta 'catch-HON-HON-PST-DECL'

(8a) is ruled out because the honorific suffix co-occurs with the *v-pst*, violating (7a); (8b) is ill-formed since the passive suffix *-ess* is attached to the *v-free* stem. This violates the constraint (7b) which requires its stem value be *v-hon-stem* or any of its subtypes. In the same vein, (8c) is not generated because the second honorific suffix occurs not with a *v-lxm*, but with a *v-hon* stem.

Within this system, the process of forming a verbal element is a step-by-step process. (9) illustrates one example:

(9) a. [[[[[cap]+usi] +ess]+ta] + ko] 'catch-HON-PST-DECL-COMP'
b. *v-lxm* → *v-hon (v-hon-stem)* → *v-tns (v-tns-stem)* → *v-free (v-stem)* → *v-comp*

As noted, we start from the type *v-lxm* and adds appropriate verbal suffixes observing the type declarations in the grammar. Notice that this means the KPSG just specifies the elements of *v-lxm*, and then the type constraints and lexical rules will form larger morphological types including *v-free* only which can appear in syntax.

2.3.2 Nominal Morphology

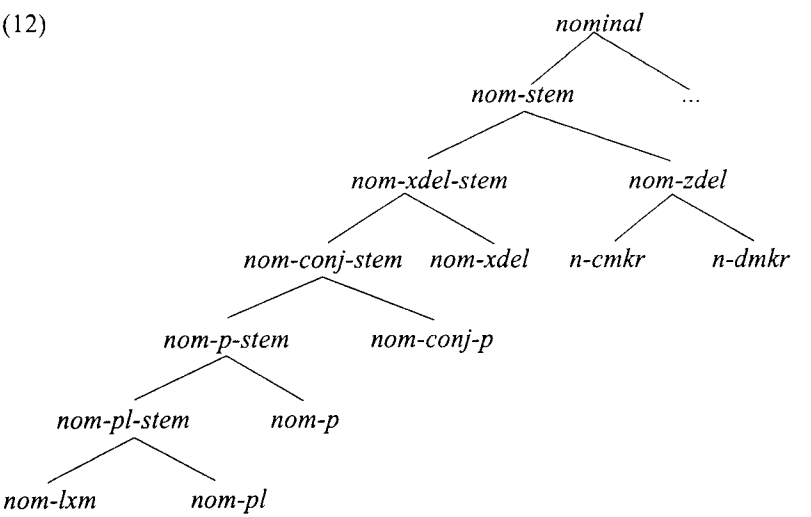
Like verbal inflections, nominal suffixes are also under tight ordering restrictions. However, all the nominal suffixes are optional as represented in the following template with one fully inflected example:

(10) N-lxm+(HON)+(PL)+(POSTP)+(CONJ)+(X-DELIM)+(Z-DELIM)

(11) sensayng+(nim)+(tul)+(eykey)+(man)+(un)
teacher+HON+PL+POSTP+X-DELIM+Z-DELIM
'to the (honorable) teachers only'

All the suffixes (often called particles) here, encoding various grammatical functions, need not be realized. Traditionally, nominal suffixes (particles) are treated as independent words even though they act more like verbal suffixes in terms of strict ordering restrictions, no intervention by any word element, and so forth. Our grammar, following lexicalist perspectives (cf. Cho and Sells 1995, Kim 1998), takes a quite different approach: we accept the view that particles do not exist as independent words but function as optional inflectional suffixes. The KPSG sets up various types of nominals depending on the attachment of particles as represented in the hierarchy (12)

(12)



These nominal stem types are defined in terms of what kind of particles appear in order as exemplified in (13):

(13) a. *nom-lxm*[sensayngnim] 'teacher'
b. *nom-pl*[sensayngnim +tul] 'teacher-PL'
c. *nom-p*[sensayngnim +tul +eykey] 'teacher-PL-DAT'
d. *nom-xdel*[sensayngnim +tul +eykey +man] 'teacher-PL-DAT-DEL'
e. *nom-xdel*[sensayngnim +tul +eykey +man +i]
'teacher-PL-DAT-DEL-NOM'

Similar to that of verbal elements, the building process of nominal elements starts from the type *nom-lxm* that includes subtypes such as *vn, n-bn, n-cn, n-cl, n-prop* (verbal nouns, bound nouns, common nouns, classifiers, proper nouns). A fully inflected nominal like (13e) thus will follow the following step:

(14) *nom-lxm* →*nom-pl-stem* →*nom-p-stem* →*nom-xdel-stem* →*nom-stem*

One crucial difference from forming a verbal element is that any of these types can directly be realized as (pumped up to) a *word* element in syntax.[6] The constraints on each type place strict ordering restrictions among nominal suffixes:[7]

(15)

$$\text{a. } nom\text{-}p \rightarrow \begin{bmatrix} \text{ORTH } \boxed{1} + \text{ eykey} \\ \text{STEM } \begin{bmatrix} nom\text{-}pl\text{-}stem \\ \text{ORTH } \boxed{1} \end{bmatrix} \end{bmatrix}$$

$$\text{b. } nom\text{-}zdel \rightarrow \begin{bmatrix} \text{ORTH } \boxed{1} + \text{ i/ka} \\ \text{STEM } nom\text{-}xdel\text{-}stem \end{bmatrix}$$

[6] The grammar specifies only *v-free* elements to be realized as *v-word* elements (subtypes of *word*) whereas for nouns it permits all the instances of type *nominal* to be realized as *n-word*. This in turn means that any subtype of *nominal* can serve as a syntactic element.

[7] Of course, the form *-eykey* is just one of the postpositions in the language, implying that the type *nom-p* is subdivided into further types.

These constraints on the nominal types can place ordering restrictions among nominal particles:

(16) a. *[$_{nom\text{-}zdel}$[sensayngnim-tul-un]-eykey] 'teacher-PL-DEL-POSTP'
b. *[$_{nom\text{-}conj}$[sensayngnim-tul-kwa]-eykey] 'teacher-PL-CONJ-Postp'
c. *[$_{nom\text{-}zdel}$[sensayngnim-tul-un]-i] 'teacher-PL-DEL-NOM'

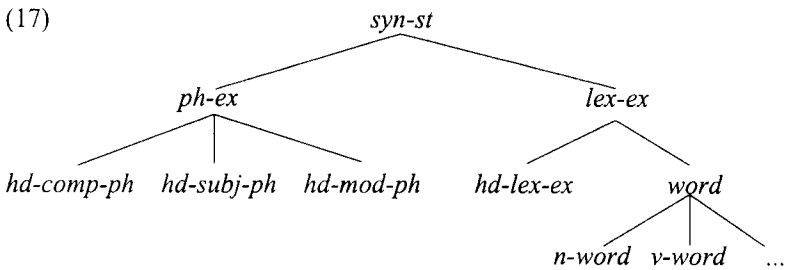
According to (15a), the so-called postposition *eykey* requires its STEM value to be an instance of *nom-pl-stem*. This explains why (16a) and (16b) are not generated in the system. (15b) ensures that the nominative marker can combine only with *nom-xdel-stem* or its subtypes: This explains why the system does not generate cases like (16c).

In sum, the morphological system we have shown makes the Korean morphology much simpler and can capture the ordering restrictions as well as cooccurrence restrictions. Other welcoming consequences of adopting the typed feature system come from the treatment of well-known mixed constructions such as sentential nominal and light verb constructions which we will discuss in section 5. Both of these have received much attention because of their mixed properties.

2.4 Syntax

Like the treatment of lexical elements, syntactic elements are classified into subtypes depending on their properties. As represented in (17), the type *syn-st* (*syntactic-structure*) is subclassified into *ph-ex* (*phrase-expression*) and *lex-ex* (*lexical-expression*):

(17)



The types in the hierarchy here represent the well-formed expressions in syntax. Put it differently, the subtypes of *syn-st* can be assumed to form the main components of Korean X-bar schema. Its two subtypes *ph-ex* and *lex-ex* are different with respect to the specifications of the LEX feature. The former is in a sense truly phrasal whereas the latter is lexical. This distinction is necessary to allow complex predicates to function as a kind of syntactic compound (see Section 4).

The type *lex-ex* has two subtypes *hd-lex-ex* and *word*. The former is for complex-predicates formed with an auxiliary verbs. The latter has subtypes such as *n-word*, *v-word*, and *adv-word*. As noted earlier, though all *nominal* elements can be projected into *n-word*, only *v-free* can be mapped (or pumped up) to *v-word* since not fully inflected stems like *mek-ess* 'eat-PST' cannot appear in syntax. One important constraint that *word* in the KPSG observes is the following Argument Realization Constraint:

(18) Argument Realization Constraint (ARC):

$$word \rightarrow \begin{bmatrix} \text{SYN}\,|\,\text{VAL} \begin{bmatrix} \text{SUBJ} & \boxed{A} \\ \text{COMPS} & \boxed{B} \end{bmatrix} \\ \text{ARG-ST} \quad \boxed{A} \oplus \boxed{B} \end{bmatrix}$$

The constraint means the elements in the ARG-ST will be realized as SUBJ and COMPS in syntax. For example, when the lexeme *ilk-* 'read'

specified only with the ARG-ST information is fully inflected as *ilk-ess-ta* 'read-PAST-DECL', its two arguments will be realized as SUBJ and COMPS as represented in (19):[8]

(19)

$$\begin{bmatrix} v\text{-}tr \\ \text{ORTH} \langle \text{ilk-} \rangle \\ \text{ARG-ST} \langle \text{NP}, \text{NP} \rangle \end{bmatrix} \rightarrow \begin{bmatrix} v\text{-}word \\ \text{ORTH} \langle \text{ilk-ess-ta} \rangle \\ \text{SYN} \begin{bmatrix} \text{HEAD} \begin{bmatrix} \text{POS } verb \\ \text{V} + \\ \text{STATIVITY} - \end{bmatrix} \\ \text{VAL} \begin{bmatrix} \text{SUBJ} \langle \boxed{1} \rangle \\ \text{COMPS} \langle \boxed{2} \rangle \end{bmatrix} \end{bmatrix} \\ \text{ARG-ST} \langle \boxed{1}\text{NP}, \boxed{2}\text{NP} \rangle \end{bmatrix}$$

As shown here, the ARC makes sure that the elements in the ARG-ST are projected into SUBJ and COMPS in syntax.

Once we have *word* elements in syntax, these elements will be combined with other syntactic elements. It is the type *ph-ex* that places restrictions on the combination of syntactic elements including *word*. This in turn means that the subtypes of *ph-ex* will tell us what kind of well-formed phrases is available in the language. The following are the grammar rules that license the subtypes of the type *ph-ex*:[9]

(20) a. Head-Subj(ect) Rule:

$$\text{XP}[hd\text{-}subj\text{-}ph] \rightarrow \boxed{1}, \mathbf{H}\left[\text{SUBJ} \langle \boxed{1} \rangle\right]$$

b. Head-COMP(lement) Rule:

$$\text{XP}[hd\text{-}comp\text{-}ph] \rightarrow \boxed{1}, \mathbf{H}\left[\text{COMPS} \langle \ldots, \boxed{1}, \ldots \rangle\right]$$

---

[8] In the lexicon, the ARG-ST value is not encoded since its information is predicted from the type *v-tr*.

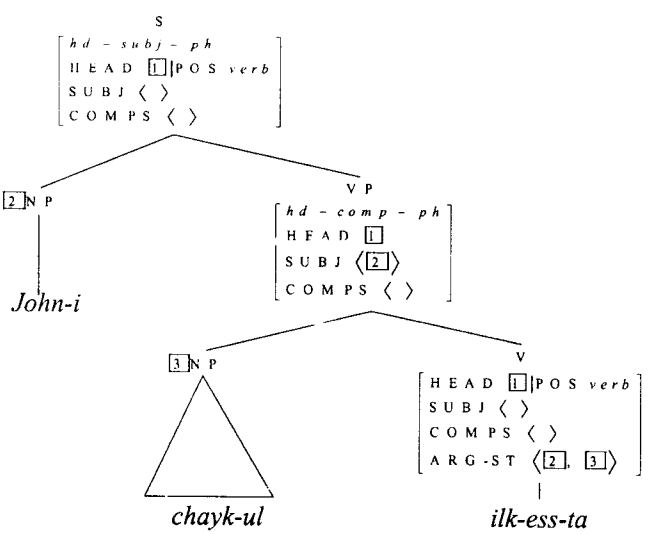
[9] In the current version of the KPSG, the grammar rules include restrictions on the case values (i.e., nominative and accusative). The space does not allow us to explicate the discussion of case phenomena in the language. See Kim (2004) for the analysis of Korean case phenomena.

c. Head-Mod(ifier) Rule:

$$XP[hd-mod-ph] \rightarrow \left[\text{MOD}\left\langle \boxed{1} \right\rangle\right], \boxed{1}\mathbf{H}$$

These simple rules can license major phrases in the language. The Head-Subj Rule, generating a *hd-subj-ph*, allows a VP to combine with its subject. The Head-COMP Rule ensures a head to combine with one of its COMPS(COMPLEMENTS) elements, forming a *hd-comp-ph*. The Head-Mod Rule allows a head to form a well-formed phrase with an adverbial element that modifies the head, resulting in *hd-mod-ph*. It is not difficult to see that these three grammar rules can eventually well-formed sentences like (21). The verb *ilk-ess-ta* 'read-PST-DECL' here selects two arguments, each of which is realized as SUBJ and COMPS according to the ARC. The head verb then combines with its COMPS *chayk-ul*, forming a well-formed *hd-comp-ph* in accordance with the Head-COMP Rule. The resulting VP then combines with the subject *John-i*, forming a *hd-subj-ph* licensed by the Head-Subj Rule.

(21)

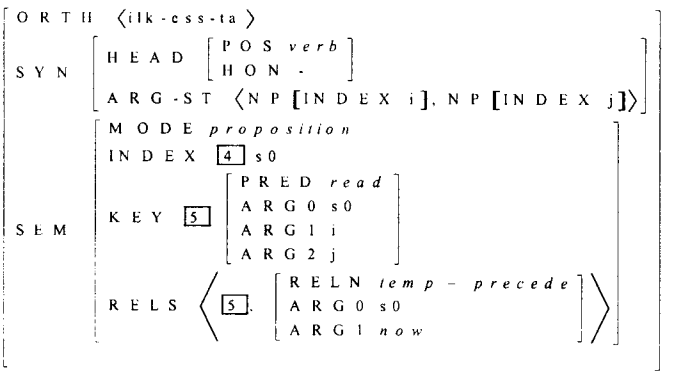


As can be observed here, the Korean X-bar grammar rules, allowing only binary structures, can capture the basic properties of Korean syntax, when interacting with other independent constraints.[10] In what follows, we will see how the framework of Korean morphology and syntax sets forth here can account for basic as well as complicated Korean syntactic phenomena.

2.5 Semantics

One main characteristics of the KPSG is that it generates the semantic structures as well as syntax. In representing the semantics, it adopts MRS (Minimal Recursion Semantics), designed to enable semantic composition using the unification of typed feature structures. It allows us to produce for each phrase or sentence a description of the meaning representation (Copestake et al. 2001). The KPSG thus includes a semantic component which is an implementation of MRS. That is, the grammar constructs the meaning descriptions by providing constraints on the feature structures that specify how a phrase's semantics is built up from the semantics of its immediate constituents (words or phrases). For example, the following is the meaning representations of a *word*:

[10] For example, the Head Feature Principle works on the type *ph-ex*, ensuring that the HEAD information of a phrase is identical with that of its head.

(22)


The MODE feature represents the semantic mode of the expression such as *proposition, question, directive* and *reference*. The value of INDEX indicates what the expression refers to. Every kind of linguistic expression refers to something that must satisfy an indicated list of restrictions for the expression to be correctly applicable. The feature RELS (RELATIONS) specifies a list of conditions that the situation or individual has to satisfy for the expression to be applicable to it. The KEY value is identified with the key semantic relation in the RELS.

When words with such semantic information combine with other syntactic (elements of *syn-st*) expressions, the meaning composition occurs in accordance with the following semantic constraints:

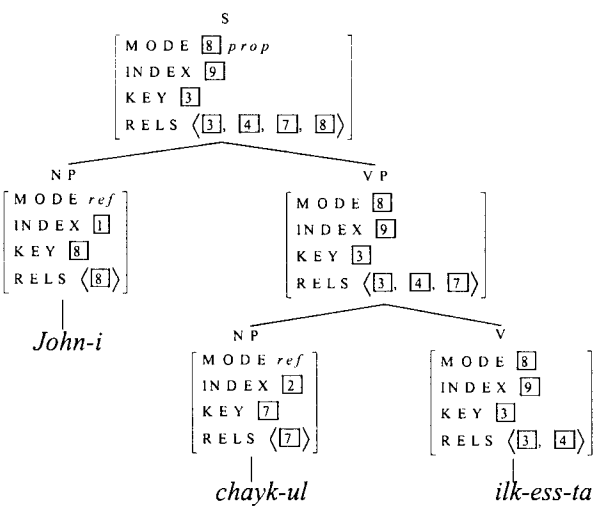
(23) a. Semantic Inheritance Principle: In any headed phrase, the mother's MODE, INDEX, and KEY values are identical to those of the head daughter

b. Semantic Compositionality Principle: In any well formed phrase structure, the mother's RELS value is the sum of the RELS values of the daughters

The following illustrates how the sentence (21) composes its

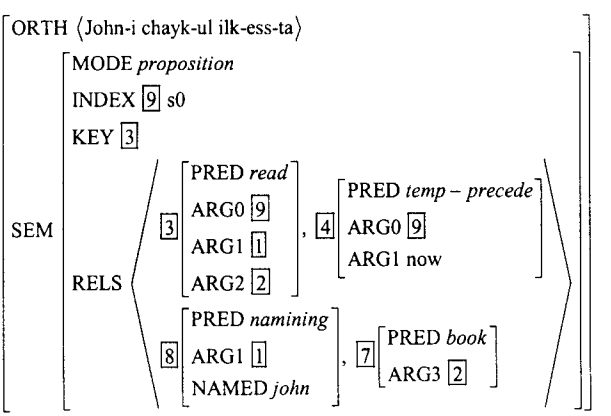
meanings in accordance with these principles:[11]

(24)



As represented in the structure, there is a situation that John read a book at a certain time in the past. The meaning of this sentence can be represented in more detail as the following:

(25)



[11] The predicate relation *temp-precede* stands for *temporarily-precede*.

If we parse this sentence in the LKB together with the implementation of this KPSG grammar, we obtain the following Figure 1 as its parsed tree and MRS meaning representation:

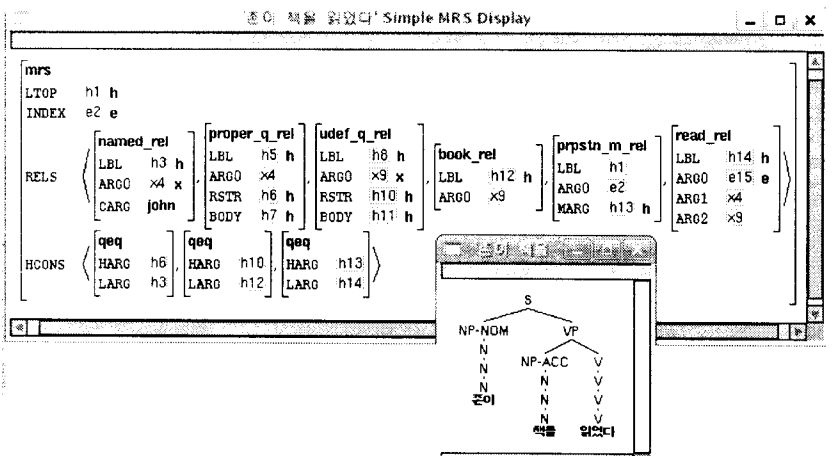


Figure 1: Parsed tree and its MRS representation

Needless to say, the sketch provided here is just the skeleton of how the semantic composition occurs within the KPSG grammar. In order to capture scope properties, it needs to introduce further features as elaborated in Copestake et al. (2003).[12] The KPSG thus not only parses syntactic structures of a given expression but also provides meaning representations that can be used for computational purposes such as machine translation.

[12] As pointed out by a reviewer, this does not show how the grammar works out for more complex phenomena such as coordination which this paper does not discuss. However, it is rather clear how the system can be extended for such phenomena, as implemented in the current version of the grammar. For example, coordination sentences will include a semantic relation such as *and-rel* whose arguments include the meaning of conjuncts. See Sag et al. (2003).

3. Major Constructions and Their Implementations

3.1 Basic Sentences with Different Argument Structures

The well-formed conditions on the subtypes of *ph-ex* can easily license basic sentence types, when interacting with the lexicon:

(26) a. [[pi-ka [o-ass-ta]]]. 'It rained.'
rain-NOM come-PST-DECL
b. [John-i [Mary-ka [silh-ess-ta]]].
John-NOM Mary-NOM dislike-PST-DECL
'John disliked Mary.'
c. [Kim-un [Mary-ka [ku chayk-ul [ilkessta-ko]]
Kim-TOP Mary-NOM the book-ACC read-COMP
[sayngkakha-ess-ta]]].
think-PST-DECL
'Kim thought that Mary read the books.'

To parse these sentences, the only thing the grammar needs to encode is the basic lexical entry of the predicates here. For example, the KPSG includes the following lexeme information for each of the predicates in (26):

(27) a. $\begin{bmatrix} v\text{-}tr \\ \text{ORTH } \langle \text{ilk-} \rangle \\ \text{SEM } read\text{-}rel \end{bmatrix}$ b. $\begin{bmatrix} v\text{-}stative\text{-}tr \\ \text{ORTH } \langle \text{silh-} \rangle \\ \text{SEM } dislike\text{-}rel \end{bmatrix}$ c. $\begin{bmatrix} v\text{-}s\text{-}tr \\ \text{ORTH } \langle \text{sayngkakha-} \rangle \\ \text{SEM } think\text{-}rel \end{bmatrix}$

These basic verb lexemes will be inflected with the tense and declarative suffixes and become *v-free* elements which then will be projected as the *v-word* elements in (26). Even though the lexical information in (27) is so simple, it will be enriched due to the fact

that each type will inherit constraints from its supertypes. For example, when the lexemes in (27b) and (27c) are fully inflected as *v-word* elements, they will at least have the following information:

(28)

a.
$$\begin{bmatrix} v-word \\ \text{ORTH} \langle \text{silh-ess-ta} \rangle \\ \text{SYN} \begin{bmatrix} \text{HEAD} \begin{bmatrix} \text{POS } verb \end{bmatrix} \end{bmatrix} \\ \text{ARG-ST} \langle \text{NP}_i, \text{NP}_j \rangle \\ \text{SEM} \begin{bmatrix} \text{MODE } prop \\ \text{INDEX s0} \\ \text{RESTR} \left\langle \begin{bmatrix} \text{RELN } dislike-rel \\ \text{ARG0 s0} \\ \text{ARG1 i} \\ \text{ARG2 j} \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix}$$

b.
$$\begin{bmatrix} v-word \\ \text{ORTH} \langle \text{sayngkakha-ess-ta} \rangle \\ \text{SYN} \begin{bmatrix} \text{HEAD} \begin{bmatrix} \text{POS } verb \end{bmatrix} \end{bmatrix} \\ \text{ARG-ST} \left\langle \text{NP}_i, \begin{bmatrix} \text{VFORM } ko \\ \text{INDEX s1} \end{bmatrix} \right\rangle \\ \text{SEM} \begin{bmatrix} \text{MODE } prop \\ \text{INDEX s0} \\ \text{RESTR} \left\langle \begin{bmatrix} \text{RELN } think-rel \\ \text{ARG0 s0} \\ \text{ARG1 i} \\ \text{ARG2 s1} \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix}$$

The ARC will ensure the elements of the ARG-ST to be realized as SUBJ and COMPS. The grammar rules then will allow each of these verbs to combine either with its SUBJ or COMP elements as discussed earlier. This will license the sample sentences in (26).

3.2 Basic Sentences with Adverbs

There are two main types of adverbs: one that can modify any verbal element (V, VP, or S), and the other that can modify only a lexical verb. The second group of adverbs include *cal* 'well', *com* 'little', *te* 'more', and *ta* 'all'. The interactions between the lexical information of adverbs and the constraints on *hd-mod-ph* are enough to generate these adverbs in right positions. For example, since *mayil* 'everyday' can modify any verbal element, all the following examples are licensed by the grammar rules:

(29) a. mayil $_{S}$[John-un haksayng-tul-eykey yenge-lul kaluchiessta]
everyday John-TOP student-PL-DAT English-ACC taught
'John taught English to students everyday.
b. John-un [mayil $_{VP}$[haksayng-tul-eykey yenge-lul kaluchi-ess-ta]].
c. John-un haksayng-tul-eykey [mayil $_{V}$[yenge-lul kaluchi-ess-ta]].
d. John-un haksayng-tul-eykey yenge-lul [mayil $_{V}$[kaluchi-ess-ta]].

Meanwhile, the second types of adverbs are lexically constrained to modify only a non-phrasal verb.

(30) a. John-i pap-ul [cal $_{V}$[mek-ess-ta]].
John-NOM meal well eat-PST-DECL
'John ate the meal well.'
b. *John-i [cal $_{VP}$[pap-ul mek-ess-ta]].

To capture these properties, the KPSG posits two subtypes *adv-ph* and *adv-lex* with their own constraints. The following is some sample feature descriptions encoding such constraints in the LKB:

```
adverbial := lexeme &
 [ SYN [ HEAD.POS adv,
         VAL [ MOD < [ SYN.HEAD.V +,
                       SEM.INDEX #index ] >,
               PRO <! !> ] ].
   SEM [ MODE none,
         INDEX event & #index ],
   ARG-ST <> ].
adv-ph := adverbial.
adv-lex  := adverbial &
 [ SYN.VAL.MOD < word & [ SYN.HEAD.AUX - ] > ].
```

These constraints then allow examples like (30a), but block those like (30b). Since the type *adv-lex* is specified to modify only a lexical word element, the grammar would not generate cases like (30b) in which the *adv-lex cal* 'well' modifies a VP.

3.3 Sentence Internal Scrambling

One thing we can notice from the grammar rules in (20) is that they generate only binary structures. This is due to the fact that the Head-COMP Rule places no restrictions on the status of the SUBJ value. It simply states that a head can combine with one of the elements in the COMPS list. The same is true of the Head-Subj Rule: it has no restriction on the COMPS value. That is, the COMPS value can be either empty or nonempty. This means that a verb can even combine with its subject before it combines with any complement.

One welcoming consequence of this system is that this binary approach then can capture sentence internal scrambling facts, one of the most complicated facts in the SOV types of language. For example, the sentence in (\ref{adv-ord}) with five syntactic elements can induce 24 (4!) different scrambling possibilities. The language allows all the following word order possibilities for (29), in addition to those given in (29):

(31) a. [mayil [haksayng-tul-eykey [John-un [yenge-lul [kaluchi-ess-ta]]]]].

b. [mayil [yenge-lul [John-un [haksayng-tul-eykey [kaluchi-ess-ta]]]]].
c. [mayil [John-un [yenge-lul [haksayng-tul-eykey [kaluchi-ess-ta]]]]].
d. ...

It is needless to say that a more desirable grammar would be one that can capture all such scrambling possibilities within minimal processing load. The KPSG grammar we sketched here requires no additional mechanism (i.e. movement operations) to allow such diverse word order possibilities. The grammar rules (the Head-Subj Rule, the Head-Comp Rule, and the Head-Mod Rule) can license and parse all these with no additional mechanisms.

3.4 Noun Phrases

Korean noun phrases are significantly different from English counterparts. One difference is that the determiner is optional:

(32) a. (ku) sakwa
the apple
b. (John-uy) sakwa
(John-GEN) apple

Another main difference concerns functions of the determiner: Unlike English, the Korean NP does not close off its phrasal projection:

(33) a. John-uy [ku chayk] '(lit.) John's this book'
John-GEN the book
b. mesci-n [John-uy [ku os]] '(lit.) fancy John's the clothes'
fancy-Rel John-GEN the clothes

As noted here in (33b), the full NP *ku os* can be modified by a genitive NP *John-uy* and then again by a modifying predicate *mesci-n*. To capture these modifier-like properties of determiners, the KPSG treats *adnominals*, the supertype of determiners, as modifiers. The actual LKB feature descriptions look like the following:

```
adnominal := lex-st &
 [ SYN [ HEAD [MOD < [ SYN.HEAD.POS noun,
                       SEM.INDEX #index ] >,
                N  -,
                V  -]]
   SEM [ MODE none,
         INDEX object & #index,
         RELS [ LIST.REST #last,
                LAST #last ] ],
   ARG-ST <> ].
```

As shown here, adnominals are modifying elements (specified in the value of MOD) whose POS value is *noun*. This specification will allow the recursive occurrence of determiners in (33) since its value just needs to bear the appropriate POS value regardless of its phrasal status.

3.5 Relative Clause Constructions

Unlike English, Korean employs no relative pronouns like *who* or *which*. In addition, the predicate of the relative clause preceding the head noun is marked with a morphological marker depending on the type of tense information (cf. Kim 1998).[13]

(34) a. Tom-i ____$_i$ ilk-nun chayk$_i$
Tom-NOM read-Pres.PN book
'the book that Tom reads'

b. Tom-i ____$_i$ ilk-un chayk$_i$
Tom-NOM read-Pst.PN book
'the book that Tom read'

c. Tom-i ____$_i$ ilk-ul chayk$_i$
Tom-NOM read-Fut.PN book
'the book that Tom will read'

The prenominal markers in \ex{0} in a sense function both as a relative pronoun and tense marker. As expected, the language also allows relativization from an embedded clause:

(35) a. John-i [Mary-ka ____$_i$ mekessta-ko] malha-n sakwa$_i$
John-NOM Mary-NOM ate-COMP say-PN apple
'the apple that John said Mary ate yesterday'

b. John-i [Mary-ka ____$_i$ ilkessta-ko] mit-nun chayk$_i$
John-NOM Mary-NOM read-COMP believe-PN book
'the book that John believes Mary read'

The key point of the KPSG treatment of relative clauses includes the type constraints on *v-mod* and gap-introducing rules. The lexical

[13] These three basic kinds of tense-sensitive prenominal markers can be extended to denote aspects when combined with tense suffixes. Thus the possible prenominal verb forms are *ilk-ten* 'read-progressive', *ilk-essten* 'read-past progressive', *ilk-essul* 'read-past conjecture', *ilk-essessul* 'read-past perfective conjecture', *ilk-ko issten* 'past perfective progressive'.

constraints on the *v-mod* will add the feature MOD to the verb with a prenominal affix, as represented in the following:

```
v-mod := v-dep.
v-rel-mod := v-mod &
 [ SYN #syn & [ HEAD [ MOOD no_mood,
                       FORM no_form ],
                VAL.MOD < ph-ex & [ SYN.HEAD.POS noun ] > ],
   SEM....
   ARGS < v-hon-stem & [ SYN #syn,
                         SEM ... ] > ].
```

This constraint states that a *v-rel-mod* element, morphologically taking a *v-hon-stem* as its STEM value (noted by the value of ARGS), will modify a nominal element.

As noted, one crucial property of relative clause constructions is that the relative clause is an incomplete sentence with one missing gap. This is captured by the gap-introducing rule. The gap introducing rule allows any of the elements in the SUBJ or COMPS to be introduced as a GAP element, as specified in the following feature description in the LKB:[14]

```
binary-start-gap-rule-1 := binary-sg &
 [ SYN.VAL [ SUBJ <>,
             COMPS <>,
             GAP <! #2 !> ],
   ARGS < #1 & [ SYN [ HEAD [ CASE.GCASE nom, PRD - ],
                       VAL [ SUBJ <>, COMPS <> ] ] ],
```

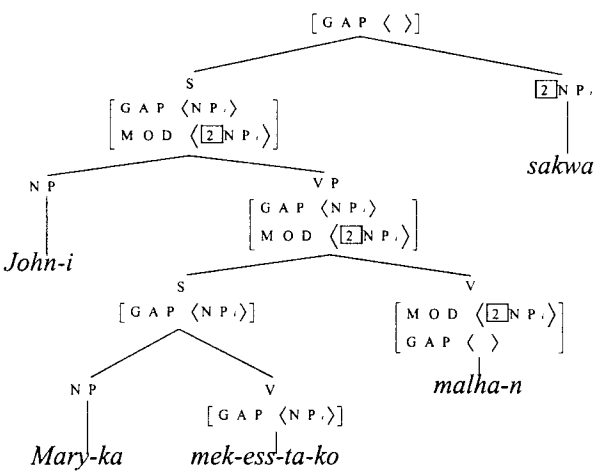
[14] The feature ARGS here means the daughter value(s) of the lefthand type. For example, this gap rule here will consist of two elements, one with the subject and the other with the head verb requiring a subject and a complement. Since the LKB recognizes only a list value (not a set value), the grammar writing requires the same number of rules.

```
[ SYN.VAL [ SUBJ < #1 >,
            COMPS < #2 > ] ] > ].
binary-start-gap-rule-2 := binary-sg & ...
```

According to the above rule, any syntactic argument can be introduced as a GAP value.[15] This GAP value is passed up to the tree until it meets its filler. The system sketched here can generate simple as well as long distance relative examples. For example, in the KPSG the example (36a) will have the structure (36b):

(36) a. John-i [Mary-ka ____$_i$ mekessta-ko] malha-n sakwa$_i$
John-NOM Mary-NOM ate-COMP say-PN apple
'the apple that John said Mary ate yesterday'

b.



[15] As noted in the ARGS value, a predicative NP cannot function as a gap as observed in the following example:

(i) a. John-i sacang-i toyessta
John-NOM president-NOM became
'John became president.'
b. *John-i toy-n sacang
John-NOM become-PNE president
'(intended) the president who became John'

Since the NP *sacang-i* here is a predicative NP, it cannot be relativized.

The word *mek-ess-ta-ko* 'eat-PAST-DECL-COMP' select two arguments. However, its COMPS can be realized as a GAP element according to the gap rule described earlier. The {\it v-rel-mod} word *malha-n* has the information that it modifies a nominal element. In addition, the relative-clause modifying rule encoded in the LKB system will terminate this GAP value when the index value of the GAP is identical with the modified nominal element:

```
head-rel-mod-rule := binary &
 [ SYN.VAL.GAP <! !>
    ARGS < ph-ex & [ SYN.VAL [ MOD < #1 & [ SYN.HEAD.POS noun,
                                            SEM.INDEX #2 ] >,
                                  GAP <! [ SEM.INDEX #2 ] !> ] ],
           syn-st & #1 & [ SYN.VAL [ GAP <! !>,
                                     ... ] ] > ].
```

As indicated in the first element of the ARGS value, the relative clause modifies a nominal element whose index value is identical with that of the GAP's value. This is how the grammar accounts for the long dependency relationship in relative clauses without resorting to any movement operations.

The system we presented here generates the following parsing tree and MRS for the above sentence.

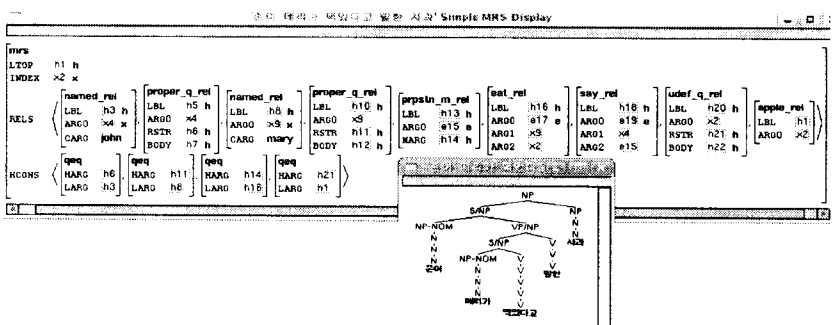


Figure 2: Parsed Tree and MRS for a Relative Clause

As can be easily observed, the grammar can correctly parse relative clauses as well as generate proper a MRS meaning representation.

4. Complex Predicate Constructions

One of the most prevalent constructions in Korean is complex predicates that consist the argument structures of two separate predicates (V2-V1) being brought together one way or another. The main constructions the grammar covers at this stage are auxiliary constructions:

(37) a. John-un sakwa-lul mek-e po-ass-ta.
John-TOP apple-ACC eat-COMP try-PST-DECL
'John tried to eat apples.'
b. John-un hangsang sakwa-lul mek-ko siph-ess-ta.
John-TOP always apple-ACC eat-COMP like-PST-DECL
'John always wanted to eat apples.'

The constructions are syntactically intriguing in that (a) it is V2 that theta-marks internal arguments and V1 thus has no influence on the number and types of arguments and that (b) the V2 takes an agentive subject but inherits its other arguments to the final predicate V1, and (c) V1 and V2 form a tight syntactic unit.

In general, the first predicate in an auxiliary construction is semantically main and the second one auxiliary displaying some aspectual and/or modality phenomena. It is required that the main verb be in a specific verb form depending on the types of auxiliary as in (38a). In addition, there is also a tight syntactic cohesion between V2 and V1: they must occur in a fixed order, always following immediately after a main verb as illustrated in (38b):

(38) a. John-i chayk-ul ilk-ko/*e siph-ess-ta.
John-TOP book-ACC read-COMP like-PST-DECL
'John wanted to read a book.'
b. *ilk-ko John-un chayk-ul siph-ess-ta.

Another main property concerns the fact that the verb does not have a normal argument structure; it is the main verb that decides the types of arguments, as shown in (39).

(39) a. John-i pyonci-lul hyucithong-ey neh-e peli-ess-ta
John-NOM letter-ACC arbage.can-LOC put-COMP do-Pst-DECL
'John has put the letter into the garbage can.'
b. John-un wul-e peli-ess-ta
John-TOP cry-COMP do-Pst-DECL
'John did the act of crying.'

Further, there is a type of auxiliary verbs that even adds a dative argument, independently of the main verb's argument structure. For example, the dative argument in (40) is added in process of forming the complex predicate.

(40) John-un Mary-eykey chayk-ul ilk-e cwu-ess-ta
John-Top Mary-Dat book-ACC read-COMP give-Pst-DECL
'John read the book to Mary.'

The KPSG attributes such basic properties to lexical information of auxiliary verbs as given in the following simplified lexical information:

(41) $\begin{bmatrix} \text{HEAD}[\text{POS } \textit{verb}] \\ \text{ARG-ST } \langle \text{NP}, [\text{LEX } +] \rangle \end{bmatrix}$

What this lexical entry means is that an auxiliary verb selects two

arguments: a subject argument and a lexical argument.

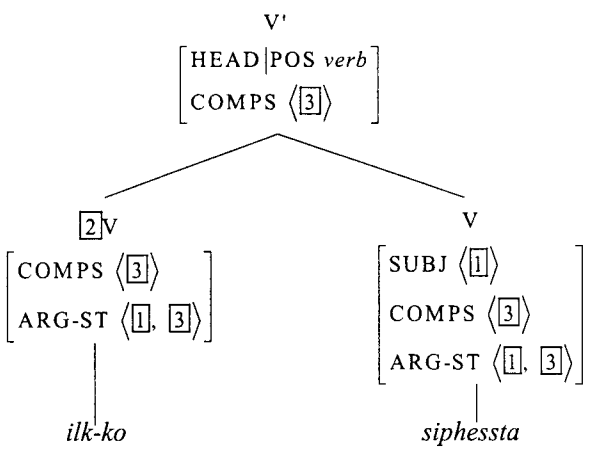
In addition to such lexical information for auxiliary verbs, the grammar introduces another well-formed condition, generating the type *hd-lex-ex* as indicated by the following Head-Lex Rule in (42):

(42) Head-Lex Rule:

$$\begin{bmatrix} hd-lex-ex \\ \text{COMPS } \boxed{A} \\ \text{LEX } + \end{bmatrix} \rightarrow \boxed{1}\begin{bmatrix} \text{LEX } + \\ \text{COMPS } \boxed{A} \end{bmatrix}, \mathbf{H}\begin{bmatrix} \text{AUX } + \\ \text{COMPS } \langle \boxed{1} \rangle \end{bmatrix}$$

The rule specifies that an auxiliary head combines with a [LEX +] complement and forms a *hd-lex-ex* which is still [LEX +].[16] Another main effect of this combination is the COMPS value of this lexical complement is passed up to the resulting phrase.[17] This rule, interacting with appropriate lexical entries for auxiliary verbs like *siph-ess-ta*, will allow a structure like the following:

(43)



[16] The feature LEX is assigned to all words and a complex predicate.

[17] This kind of argument composition is different from the previous analyses (cf. Bratt 1996, Chung 1998, Kim 2000), mainly in that the composition happens in syntax rather than in the lexicon.

The auxiliary verb *siphessta* 'would-like' takes two arguments: one realized as a subject and the other as a complement. When the auxiliary combines with the main verb, the result forms a *hd-lex-ex* and inherits the main verb's COMPS value in accordance with the Head-Lex Rule.

The soundness of the approach is proved by the parsed tree for the sentence (38a) and its MRS representation in Figure 3. As shown in the parsed tree, the main verb and the auxiliary form a complex predicate. This complex predicate, inheriting the COMPS value of the main verb, thus still requires its COMPS and SUBJ. The MRS also represents there is a situation where the person named 'John' would like to read a book.

5. Mixed Category Constructions

Within the KPSG, as noted earlier, all the linguistic expressions are types of *sign* and these types are organized according to their properties with the multiple inheritance mechanism. This is the key to account for the mixed properties of phenomena such as gerundive and light verb constructions:

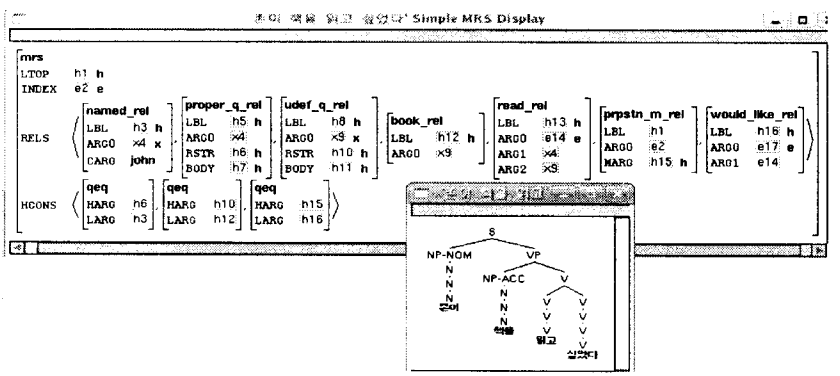
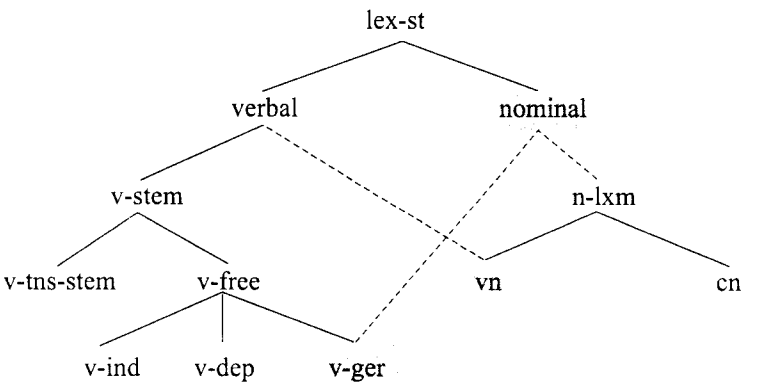


Figure 3: Parsed Tree and MRS for a Auxiliary Sentence

(44) a. sensayngnim-i chayk-ul ilk-usi-ess-um.
teacher-NOM book-ACC read-HON-PAST-NMLZ
'The teacher has read the book.'
b. hankwuk-i catongcha-lul mikwuk-ey swuchwul-ul hayessta
Korea-NOM cars-ACC America-LOC export-ACC did
'Korea exported really a lot of cars to America.'

The following is a simplified hierarchy, representing the relevant part:[18]

(45)



As we can notice here in the hierarchy, the type *vn* and *v-ger* are declared to be the subtypes of both verbal and nominal. This in turn

[18] As a reviewer questioned, one can wonder whether this type hierarchy satisfies the so-called unique greatest lower bounds (glb) condition (for two types to be compatible, they must have a unique highest common descendent). One thing to notice here is that the dot line here means the existence of other types between the two types. For example, there exist types such as *n-p-stem* and *n-xdel-stem*, each of which have particles attached to. In writing a grammar with various (not random but linguistically motivated) types, grammar writers cannot specify every glb type: one merit of the LKB system is that it automatically generates a glb type when the condition does not meet. (cf. Copestake 2002: 41-43). In the actual implementation of the grammar in the LKB, we observe one glb type between *verbal* and *vn* and *v-ger*.

means that the types *vn* and *v-ger* will inherit all the constraints of these supertypes some of which are given in the following:

(46) a. nominal: [N +]

b. verbal: $\begin{bmatrix} \text{V} + \\ \text{ARG-ST} \langle [\ ], \ldots \rangle \end{bmatrix}$

c. v-stem: [POS *verb*]

d. n-lxm: [POS *noun*]

Our grammar, differing from standard approaches, thus introduces three HEAD features POS, V, and N. All instances of the type *verbal* will have the feature [V +] and have non-empty ARG-ST values. This in turn means that the major categories are distinguished by the usual features N and V. In addition, the grammar assigns [POS *verb*] to the type *v-stem* and [POS *noun*] to the type *n-lxm*.

Within these constraint specifications, the inheritance mechanism will eventually ensure the types *vn* and *v-ger* to have at least the following information inherited:

(47)

a. *vn* →
$$\begin{bmatrix} \text{SYN}|\text{HEAD} \begin{bmatrix} \text{POS } noun \\ \text{N} + \\ \text{V} + \end{bmatrix} \\ \text{ARG-ST} \langle [\ ], \ldots \rangle \\ \text{SEM} \ \ldots \end{bmatrix}$$

b. *v-ger* →
$$\begin{bmatrix} \text{SYN}|\text{HEAD} \begin{bmatrix} \text{POS } verb \\ \text{N} + \\ \text{V} + \end{bmatrix} \\ \text{ARG-ST} \langle [\ ], \ldots \rangle \\ \text{SEM} \ \ldots \end{bmatrix}$$

For example, let us see how the grammar forms a gerundive verb. In forming a gerundive verb, the KPSG starts from a transitive verb

lexeme *ilk-* 'read' given in (48a) and then forms a *v-tns-stem* with the attachment of the past tense suffix *-ess*. This *v-tns-stem* then combines with the nominalizer suffix *um*, forming the type *v-ger*. In this process, verbal properties (e.g. POS and V values) are inherited from *v-lxm* and *v-tns-stem*, whereas their nominal properties (e.g. N) are incurred from its supertype *nominal*. The gerundive verb *ilk-ess-um* 'read-PST-NMLZ' will thus have at least the lexical information given in (48b):

(48)

a.
$$\begin{bmatrix} v\text{-}lxm \\ \text{ORTH ilk-} \\ \text{SYN}\begin{bmatrix} \text{HEAD}\begin{bmatrix} \text{POS } verb \\ \text{V} + \\ \text{N} - \end{bmatrix} \\ \text{ARG-ST } \langle \boxed{1}\text{NP}, \boxed{2}\text{NP} \rangle \end{bmatrix} \end{bmatrix}$$

b.
$$\begin{bmatrix} v\text{-}lxm \\ \text{ORTH } [\text{ilk} + \text{ess}] + \text{um} \\ \text{SYN}\begin{bmatrix} \text{HEAD}\begin{bmatrix} \text{POS } verb \\ \text{V} + \\ \text{N} + \end{bmatrix} \\ \text{ARG-ST } \langle \boxed{1}\text{NP}, \boxed{2}\text{NP} \rangle \end{bmatrix} \end{bmatrix}$$

Such cross-classification of the type *v-ger*, allowing multiple inheritance, is also reflected in the feature descriptions in the LKB. The following represents a sample source code:

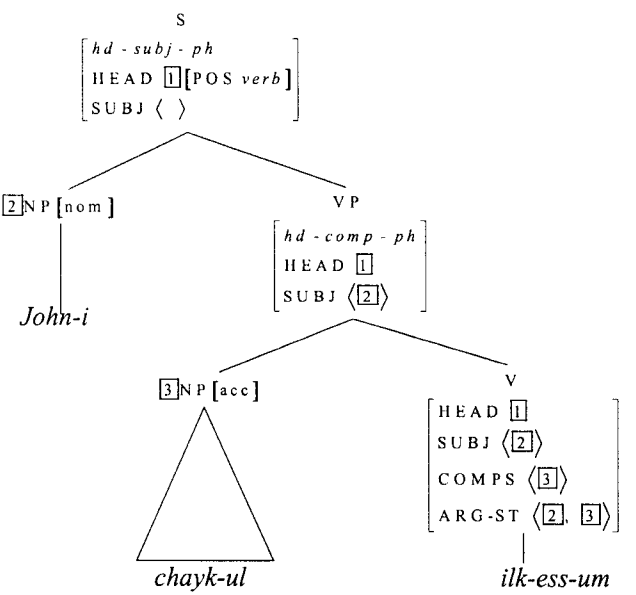
```
v-ger := v-free & nom-pl-stem &
 [ SYN #syn & [ VAL.MOD <> ],
   SEM #sem,
   ARG-ST #argst,
   ARGS < v-tns-stem & [ SYN #syn, SEM #sem, ARG-ST #argst ] > ].
```

As observed here, as a subtype of *v-free* and *n-lxm*, the type *v-ger* thus inherits the constraints from its supertypes. Being a subtype of *v-free*, it inherits verbal properties from the type *v-free*, selecting arguments and assigning case values to them. Since it is a subtype of *nominal*,

v-ger would undergo nominal suffixation processes such as case attachment. In addition, the grammar introduces the binary-valued features V on the type *verbal* and N on the type *nominal*, which play crucial roles in capturing mixed properties as well as various generalizations in the Korean grammar.

Once we form a gerundive verb, we then now can see how it appears in syntax and participate in forming a bigger phrase. For example, the gerundive verb in (48b) will be projected into a structure like (49):

(49)



As noted, the gerundive verb *ilk-ess-um* 'read-PST-NMLZ' inherits from its lexeme *ilk-* 'read' the projects such as argument structure value and head features. This explains why the gerundive verb selects a nominative subject, can be modified by an adverb, combines with the sentential negative marker, occurs with an auxiliary verb, and the

like. Because the gerundive verb selects the same argument(s) as the verb lexeme it is derived from, it forms a well-formed *hd-comp-ph* when it combines with its argument *chak-ul*. This resulting VP combines with the SUBJ, forming a *hd-subj-ph*, just like a sentence. This is what the top node S in (49) represents, reflecting the internal properties of GPs.

Strong support for the treatment of the gerundive predicate as a projection of [POS *verb*] can be found from (a) the presence of a tense and an agreement suffix and (b) the possibility of heading an independent sentence as given in (50):

(50) sensayngnim-i chayk-ul **ilk-usi-ess-um**.
teacher-NOM book-ACC read-HON-PAST-NMLZ
'The teacher has read the book.'

The example in (50) is just like a sentence with the mood marking *-ta* instead of the gerundive marking *-um*, except for some nominal properties as discussed.

The analysis also provides a simple way of capturing relativization and extraction phenomena. Though GPs externally act like noun phrases (because of the N feature), they do not serve as the head of a restrictive relative clause as repeated here in (51).

(51) *John-un [[salam-tul-i molulila-ko sayngkakha-n] [Mary-ka
John-TOP people-PL not.know-COMP think-REL Mary-NOM
ilccik ttenass-um]]-ul alassta.
early left-NMLZ knew
'*John knew [Mary's leaving early] that he thought that people wouldn't notice'.

The only thing the grammar needs to specify is the constraint that a relative clause modifies a phrase projected from the feature [POS *noun*] not the one with [N +]. This is what we can find in the 'head-rel-mod-rule' given in section 3.5. As indicated in the first element of the ARGS value in this rule, the relative clause can modify an element whose head bears [POS *noun*] value. This will then correctly block any relative clause from modifying a gerundive verb head though it behaves like a nominal element.

It is possible to extract an element from GPs, indicating that the GP behaves more like Ss and less like NPs in terms of the external syntax:

(52) **ku chayk-un** na-nun [John-i ____ ilkess-um]-ul mitnunta
that book-TOP I-TOP John-NOM ____ read-NMLZ-ACC believe
'That book, I believe John read.'

This is unexpected when considering the external status of the GP to be a pure NP. The KPSG, allowing extraction from a sentence level element ([POS *verb*]), takes the GP to be just like a sentence with the positive N feature. Nothing thus blocks this extraction as proven by one of the parsed trees and its MRS representation in the Figure 4.[19] As noted in the parsed tree, the object of the embedded gerundive *ilkess-um* is dislocated to the sentential initial position. The grammar can correctly link these two. An analysis in which the gerundive verb is taken to be either simply a verb or simply a noun, cannot make such predictions in a simple and straightforward way.

[19] The present grammar generates several parsed trees for such a sentence because of the ambiguity of the *-nun* marked NPs. The *nun* marked NP can be interpreted either as a dislocated topic NP or a contrastive topic marker in situ.

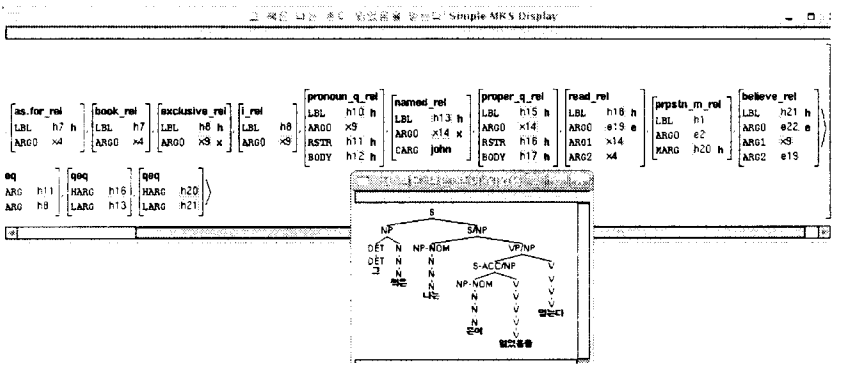


Figure 4: Parsed Tree and MRS for a Gerundive Sentence

6. Testing the Feasibility of the System

The grammar we have built within the typed-feature structure system here, eventually aiming at working with real-world data, has been first implemented into the LKB. In testing the performance and feasibility of the grammar, we first built our test sets from (1) the SERI Test Suites '97, (2) the Sejong Project Basic Corpus, and (3) self-constructed examples adopted from the literature. The SERI Test Suites (Sung and Jang 1997), designed to evaluate the performance of Korean syntactic parsers, consists of total 472 sentences (292 test sentences representing the core phenomena of the language and 180 sentences representing different types of predicate). Meanwhile, the Sejong Corpus have 179,082 sentences with about 2 million words. We selected simple sentences (the average number of words in each sentence is about 5) from the corpus that show the phenomena the grammar tries to account for. The self-constructed examples, consisting of 350 sentences, are mainly drawn from the literature that cannot be easily found in the Sejong or SERI suites. For example, sentential

internal scrambling cases, long distance relative clause cases, gerundive constructions, and so forth are not in the two suites, but are often discussed in the literature. Since the aims of our project is to build a system that allows deep parsing, we have tried to include most of the sentences and phenomena discussed in the relevant literature.[20]

The following table shows the simple results of parsing the selected items from the constructed test suites:

(57)

Corpus Types	# of Ss	# of Test Ss	# of Parsed Ss
SERI Test Suite	472	472	443 (93.8%)
Sejong Corpus	179,082	200	178 (89%)
Self-designed Test Suite	350	350	333 (95%)

As the table shows, the parsing results of its implementation are quite positive. In terms of parsing the SERI test suites that include the major linguistic examples that have been discussed in literature, it parsed 443 sentences out of total 472. Failed sentences are related to the grammar that the current system has not yet written. For example, the SERI Test Suites include examples representing phenomena such as honorification, coordination, and left dislocation of subject. However, in parsing the Sejong Corpus includes authentic examples, the grammar has lower parsing rates. This is partly from the complexity of the authentic data which the project needs to advance

[20] As a reviewer points out, we admit that we need to have better-designed and comprehensive suites. Though this is partly our fault, it is at this point hard to find any reliable test suits for Korean parsers. In addition, in a grammar writing project, we cannot but start with simple and core sentences that show the main properties of the language. Once we set up a more reliable grammar, we plan to expand the current test suites.

more. It is believed that once we have a finer-grained grammar for these phenomena, the KPSG will resolve these remaining sentences.

As a reviewer pointed out, the simple results shown here cannot show us whether this system will bring high performance and can be extended to large scale of data.[21] However, the results here at least can tell us it is possible to have a deep grammar for Korean, which has often been neglected in the research of Korean parsing systems.

7. Conclusion

It is hard to deny the fact that in building an efficient grammar, expressive accuracy has often been sacrificed in order to achieve computational tractability (Oepen et al. 2002). However, putting linguistic generalizations aside has brought difficulties expanding the coverage and eventually building a large scale of grammar. The morphological, syntactic, and semantic parsing system we have developed here with typed feature structures is an effort to solve such preexisting problems while keeping linguistic insights, thus making the Korean morphology, syntax, and semantics much simpler.

The work described here, even though it is an on-going project, achieves welcoming coverage of major constructions that have been widely discussed in the literature (but cannot often be found in real data), providing a promising future direction. The research presented in this paper provides robust parsing results, asking for the further development of this system to build a much more efficient, reliable Korean syntactic as well as semantic parser than existing ones. We

[21] One thing we need to remember is that such a question follows from almost all the projects that aims for deep processing. A trade-off between deep and shallow processing is thus in a sense unavoidable. See Butt and King (2003) for the discussion of this issue.

hope to have shown that the ongoing project of developing the KPSG and its implementation into the LKB system can be said to be interesting in two respects: as a means of testing linguistic hypotheses and seeking the potential to be integrated in applications which require natural language understanding, including machine translation, question-answering system, NL interfaces, and so forth.

One final word in order is that, as pointed out by Butt and King (2003), grammar writing is a difficult and underappreciated task in natural language processing, since it requires both solid linguistic instincts as well as a deep understanding and interest in the computational aspects of language processing. We hope this report can motivate further research in Korean grammar writing for the lagging development in this area, compared to the research for other languages such as English and Japanese.

References

Butt, Mirium and Tracy King. 2003. Grammar Writing, Testing, and Evaluation. In Ali Farghaly (ed.), *Handbook for Language Engineers*, 129-180. CSLI Publications.

Bratt, Elizabeth Owen. 1996. *Argument Composition and the Lexicon: Lexical and Periphrastic Causatives in Korean.* Doctoral dissertation, Stanford University.

Cho, Young-Mee Yu, and Peter Sells. 1995. A lexical account of inflectional suffixes in Korean. *Journal of East Asian Linguistics* 4, 119-174.

Chung, Chan. 1998. Argument Composition and Long-Distance Scrambling in Korean: An Extension of the Complex Predicate Analysis. In Hinrichs et al. (eds) *Complex Predicates in*

Nonderivational Syntax, 159-220. New York: Academic Press.

Copestake, Ann, Dan Flickinger, Ivan Sag and Carl Pollard. 2001. Minimal Recursion Semantics: An introduction. Ms. Stanford University.

Copestake, Ann and Flickinger, Dan. 2000. An Open-source Grammar Development Environment and Broad-coverage English Grammar Using HPSG. In *Proceedings of the Second Conference on Language Resources and Evaluation*, 591-600.

Copestake, Ann. 2002. *Implementing Typed Feature Structure Grammars*. CSLI Publications.

Kang, Seung-Shik. 1999. The Status of the Korean Language Processing and Future Directions (In Korean). In *Proceedings of the 6th Korean Phonetics Association Conference*.

Kim, Jong-Bok. 1998. Interface between Morphology and Syntax: A Constraint-Based and Lexicalist Approach. *Language and Information* 2: 177-233.

Kim, Jong-Bok. 2000. *The Grammar of Negation: A Constraint-Based Approach*. Stanford: CSLI Publications.

Kim, Jong-Bok. 2004. *Korean Phrase Structure Grammar*. Hankook Publishing.

Kim, Jong-Bok and Jaehyung Yang. 2003. Korean Phrase Structure Grammar and Implementing it into the LKB System (In Korean). *Korean Linguistics* 21: 1-41.

Kim, Jong-Bok and Jaehyung Yang. 2004. Projections from Morphology to Syntax in the Korean Resource Grammar: Implementing Typed Feature Structures. In *Lecture Notes in Computer Science* Vol 2945, pp. 13-24, Springer-Verlag.

Oepen, Stephan, Dan Flickinger, Jun-ichi Tusujii, and Hans Uszkoreit. 2002. *Collaborative Language Engineering*. CSLI Publications.

Pollard, Carl and Ivan Sag. 1994. Head-driven Phrase Structure Grammar. Stanford: CSLI Publications.

Sag, Ivan, Tom Wasow, and Emily Bender. 2003. *Syntactic Theory: A*

Formal Approach. Stanford: CSLI Publications.

Shim, Kwangseob and Yang Jaehyung. 2002. MACH: A Supersonic Korean Morphological Analyzer. In *Proceedings of Coling-2002 International Conference*, pp. 939-45, Taipei.

Siegel, Melanie and Emily M. Bender. 2002. Efficient Deep Processing of Japanese. In Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization. Coling 2002 Post-Conference Workshop. Taipei, Taiwan.

Siegel, Melanie. 2000. HPSG Analysis of Japanese. In: W.Wahlster(ed.), *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer Verlag.

Sung, Won-Kyung and Myung-Gil Jang. 1997. SERI Test Suites '95. In *Proceedings of the Conference on Hanguel and Korean Language Information Processing*.

Jong-Bok Kim
School of English
Kyung Hee University, 130-701
jongbok@khu.ac.kr

Jaehyung Yang
School of Computer Engineering
Kangnam University, Kyunggi, 446-702
jhyang@kangnam.ac.kr

Received: April 24, 2006
Revised version: June 21, 2006