Korean Phrase Structure Grammar and Its Implementations into the LKB System

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

Jaehyung Yang
Kangnam University
School of Computer Engineering
jhyang@kangnam.ac.kr

1 Introduction

Though there exist various morphological analysers developed for Korean, no serious attempts have been made to build its syntactic or semantic parser(s), partly because of its structural complexity and partly because of the existence of no reliable grammar-build up system. This paper presents a result of our on-going project to build up a computationally feasible Korean Phrase Structure Grammar (KPSG) and implementing it into the LKB (Linguistic Knowledge Building) system.

The grammatical framework we adopt for KPSG is the constraint-based grammar, HPSG (Pollard and Sag 1994, Sag and Wasow 1999). The grammar HPSG (Sag and Wasow 1999) is well suited to the task of multilingual development of broad coverage grammars. 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 facilities the extension of grammar in a systematic and efficient way, resulting in a linguistically precise and theoretically motivated descriptions of Korean. In addition, we adopt a flat semantic formalism Minimal Recursion Semantics (MRS) in representing semantics (Copestake et al. 2001). MRS is proved to be flexible and well work with the Korean typed feature structures too.

The basic tool for writing, testing and processing the KPSG is the LKB system (downloadable from http://www-csli.stanford.edu/ aac/lkb.html, Copestake 2002). The LKB system is a grammar and lexicon development environment for use with constraint-based linguistic formalisms such as HPSG.

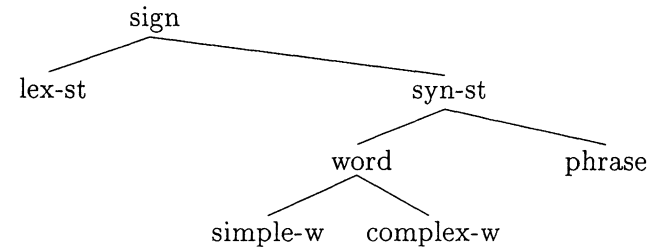
2 Korean Phrase Structure Grammar

KPSG is basically an extension of the constraint based grammar, HPSG. HPSG is built upon a nonderivational, constraint-based, and surface-oriented grammatical architecture. Though HPSG shares with the P&P (Principles and Parameters) the idea that interaction between lexical entries and a set of parameterized principles determines grammatical well-formedness, it has one fundamental architectural difference from the P&P framework: there are no derivational or transformational operations involved. Unlike the P&P framework where distinct levels of syntactic structure are sequentially derived by means of the transformational operation Move-α (affecting both phrasal categories and heads), HPSG has no notion of deriving one structure from another structure. It employs a concrete conception of constituent structures, a limited set of universal principles (e.g. the Head Feature Principle, the Valence Principle, etc.), and enriched lexical representations.

The Korean Phrase Structure Grammar (henceforth KPSG) consists of grammar rules, inflec-

tion rules, lexical rules, type definitions, and lexicon. All the linguistic information is represented in terms of signs. These signs are classified into subtypes as represented in a simple hierarchy in (1):

(1)



The elements in *lex-st* type, the basic components of the lexicon, are formed from either lexicon or lexical rules and then can serve as input to syntax. In what follows, we will first consider how the system builds such lexical elements.

2.1 Building up a word and the structure of lexicon

Korean is an agglutinative langauge with a very productive inflectional system. One example of its verb inflectional system could tell us its complexity (cf. Cho and Sells 1995, Kim 1998b):

(2) cap + hi + si + ess + kess + ta
V-root + (Pass/Caus) + (Hon) + (Tns) + (Asp) + Decl

As given in (2), the suffixes cannot be attached arbitrarily to a stem or word, but have a regular fixed order. In addition, all the verbal suffixes are optional except the mood marker. That is, for a verb stem to appear in syntax, it should be inflected at least with a mood marker (cf. Kim 1998b). In order to handle such possible ways of combining inflections, KPSG subclassifies *verb-lexeme* into two subtypes *v-stem* and *v-free*: only verbs belonging to the latter can appear in syntax. The further subclassifications of these two types are as follows:[1]

(3) a. v-lexeme: v-stem, v-free
b. v-stem: v-base, v-bound
c. v-bound: v-hon, v-tense
d. v-free: v-mod, v-ind, v-comp

KPSG, equipped with inflectional lexical rules, builds up correct verb forms including the *v-free* elements that can function as inputs in syntax.

Noun inflections are quite different the verb, in that any noun stem can appear in syntax, as represented in (4):

(4) sensayng + (nim) + (tul) + (eykey) + (man) + (un)
teacher + Hon + Pl + Postp + Del + Top
'to the (honorable) teachers only'

All the suffixes (often called particles) here are optional. The adopted type classification allows any noun stem to function as a syntactic element, Unlike the Japanese grammar developed by Siegel and Bender (2002) for the LKB system, KPSG treats these particles as suffixes.

In KPSG, each lexical entry is thus fully inflected and words are thus represented by feature structures containing orthographic, syntactic, and semantic information. A properly inflected verbal or nominal element is then projected into syntax with the interactions of well-formed phrase constraints in syntax. The following description represents a minimized information on the type of *v-tr* and a sample verb in the grammar:

[1] *v-mod* words are prenominal verbs, *v-ind* words are verbs with declarative, imperative, and suggestive markings, and *v-comp* words include those ended with a complementizer form.

```
v-tr := v &
 [ SYN.ARG-ST < phrase & [ SYN.HEAD.CASE nom ],
                phrase & [ SYN.HEAD.CASE acc ] > ].

cap := v-np-tr &
[ ORTH.LIST.FIRST "cap",
  SEM "catch_rel" ].
```

2.2 Syntax

All the syntactic rules in KPSG are either unary or binary. Different from English (and from the Japanese grammar of Siegel and Bender 2002, Siegel 2000), we assume that Korean adopts the following phrasal well-formed conditions:

(5) Korean X′ Syntax

```
a. hd-arg-ph:
 [ ] -> #1, H[ARG-ST <⋯#1⋯>]
b. hd-mod-ph:
 [ ] -> [MOD #1], H[#1]
c. hd-filler-ph:
 [ ] -> #1, H[GAP <#1>]
d. hd-word-ph:
 [word] -> [word], H
```

(5)a means that when a head combines with one of its arguments, the resulting phrase is a well-formed phrase. (5)b allows a head to combine with a phrase that modifies it. (5)c is a constraint for a head to form a phrase (with a missing a gap) with a filler. (5)d basically generates a word level syntactic element by the combination of a head and a word. This well-formed phrase condition, not found in languages like English, forms various types of complex predicates found in the language. The simple X′ syntax, whose motivations we will see in due course, can capture the major syntactic structures of Korean in a straightforward manner.

3 Major Korean Constructions and Implementations

3.1 Basic Sentences

The well-formed conditions of *head-arg-ph* can easily license basic sentence types:

(6) a. [[pi-ka [o-ass-ta]]]. 'It rained.'
rain-NOM come-Past-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 [ilk-ess-ta-ko]] [sayngkakha-ess-ta]]].
Kim-Top Mary-Nom the book-Acc read-Pst-Decl-Comp think-Pst-Decl
'Kim thought that Mary read the books.'

Since the phrase condition allows a head (lexical or phrasal) to combine only with one syntactic argument, KPSG generates only binary structures as represented by the brackets.

This binary approach then allows efficient structure parsing by capturing sentence internal scrambling facts, one of the most complicated facts in SOV types of language. For example, the sentence in (7) with five syntactic elements can induce 24 (4!) different scrambling possibilities.

(7) mayil John-un haksayng-tul-eykey yenge-lul [kaluchi-ess-ta]
Everyday John-Top students-Pl-Dat English-Acc teach-Past-Decl
'John taught English to the students everyday.

A most effective grammar would no doubt be the one that can capture all such scrambling possibilities within minimal processing load. In KPSG, the condition on *hd-arg-ph* written in three rules, one of which is given in the below, can serve this function:[2]

```
head-arg-rule-1 := hd-arg-ph &
 [ SYN.ARG-ST #2,
   ARGS < #1,
          syn-str & [ SYN.ARG-ST [ FIRST #1,
                                   REST #2 ] ] > ].
```

3.2 Basic Sentences with Adverbs

There are at large 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', *ta* 'all', etc. The interactions between the lexical information of adverbs and the constraints on *head-mod-ph* are enough to generate these adverbs in right positions. For example, since *mayil* 'everyday' can modify any verb syntactic element, KPSG processes the following modification alternatives for (7):

(8) a. mayil $_S$[John-un haksayng-tul-eykey yenge-lul kaluchi-ess-ta]].
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 verb element.

(9) a. John-i pap-ul [cal $_V$[mek-ess-ta]].
John-Nom meal well eat-Past-Decl
'John ate the meal well.'

b. *John-i [cal $_{VP}$[pap-ul mek-ess-ta]].

To capture these properties, KPSG posits two subtypes *adv-phmod* and *adv-wmod* with their own constraints:

```
adverbial := lexeme &
 [ SYN [ HEAD adv & [ MOD < [ SYN.HEAD verb,
                              SEM.INDEX #index ] > ],
         VAL [ ARG-ST <>,
               PRO <! !> ] ],
   SEM [ INDEX event & #index,
         RELS [ LIST.REST #last,
                LAST #last ] ] ].

adv-phmod := adverbial.

adv-wmod  := adverbial &
 [ SYN.HEAD.MOD < simple-w & [ SYN.HEAD.AUX - ] > ].
```

[2]Since the LKB does not allow a set operation, the LKB implementation requires to write three head-arg-rules depending on which argument in the ARG-ST combines with the head.

This system then allows examples like (9)a, but blocks those like (9)b.[3] Since adverbs like *cal* 'well' here are lexically specified to modify only a lexical element, the grammar would not generate cases like (9)b.

3.3 Case

Case also reflects a major characteristic of the language. Given a proper context, case markers can freely be omitted or replaced by delimiters:

(10) a. John-(i) Mary-man-(ul) manna-ass-ta
John-Nom Mary-only-Acc meet-Pst-Decl
'John met only Mary.'

b. John-un Mary-man manna-ass-ta

c. John Mary manna-ass-ta

Unlike previous approaches (cf. Seigel 2000), KPSG treats case markings as a kind of inflection (Cho and Sells 1995, Kim 1998b). The mechanism of unification and type systems in KPSG function as the main basis in parsing the case omission and relevant facts in a straightforward manner. The grammar starts with the classification of nouns into several types according to its case markings as in (11):

(11) n-word: n-nom-w, n-acc-w, n-dat-w, n-gen-w, n-unk-w
n-unk-w: topic-w, delimiter-w

The type *n-nom-w* is specified to have the head feature [CASE *nom*] whereas *n-unk-w* with [CASE *case*] in which *case* is the supertype all the case values:

```
n-nom-w := n-word & [SYN.HEAD [ CASE nom, D-MKR no-mkr] ].
n-unk-w := n-word & [SYN.HEAD [ CASE case] ].
```

For example, a verb like *manna-* 'meet' will select a nominative subject and an accusative NP as in (12)a.

(12) ARG-ST < NP[*nom*], NP[*acc*]>

Since the feature values [*nom*] and [*acc*] are subsumed by its supertype *case*, all the case values in (10) are appropriate unifications. Equipped with the *head-arg-ph*, the grammar further generates two structures for (10b) and (10c), depending on the grammatical function of the non-case marked NP.

3.4 Noun Phrases

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

(13) a. (ku) sakwa
the apple

b. (John-uy) sakwa
(John-Gen) apple

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

[3]The phrase condition that allows to form such a combination is *head-word-ph*.

(14) 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 (14)b, the full NP can be recursively modified by a genitive NP and then by a modifying predicate. To capture these modifier-like properties, KPSG treats determiner phrases as modifiers. Even a genitive noun is specified to modify a nominal element as represented in the following:

```
n-gen-w := n-word &
 [ SYN.HEAD [ CASE gen, MOD < [ SYN.HEAD noun ] > ] ].
```

3.5 Relative Clause

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 1998a). [4]

(15) 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
Tom$_i$ read-Fut.PN book
'the book that Tom will read'

The prenominal markers in (15) in a sense function both as a relative pronoun and tense marker. As expected, the language also allows a relativization from an embedded clause:

(16) 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 includes the type constraints on *v-mod-w* and gap-introducing rules. The lexical constraints on the *v-mod-w* will add the head feature MOD to the verb with a prenominal affix, as represented as follows:

```
v-mod := v-dep &
 [ SYN #syn & [ HEAD.MOD < [ SYN.HEAD noun ] > ],
   SEM.RELS [ LIST [ FIRST [ ARG0 #s,
                             ARG1 #u ],
```

[4] 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'

```
                    REST.FIRST [ PRED "now",
                                 ARG0 #u ],
                    REST.REST #list ],
              LAST #last ],
   ARGS < v-stem2 & [ SYN #syn,
                      SEM [ INDEX #s,
                            RELS [ LIST #list,
                                   LAST #last ] ] ] > ].
```

Meanwhile, the gap introducing rules allow any of the elements in ARG-ST to be introduced as GAP element, as given in the following:

```
binary-start-gap-rule-1 := binary-sg &
 [ SYN.VAL [ GAP <! #1 !>,
             ARG-ST <   > ],
   ARGS < #2 & [ SYN.HEAD.PRD - ],
          syn-st & [ SYN.VAL.ARG-ST < #1, #2 > ] > ].

binary-start-gap-rule-2 := binary-sg ...
binary-start-gap-rule-3 := binary-sg ...
```

According to the above rule, any syntactic argument can be introduced as a GAP value and this GAP value is passed upto the tree until it meets the filler. This system can generate simple as well as long distance relative examples.

The present grammar further can parse so called genitive relative clauses like (17)a. Another welcome result of the grammar is generating two syntactic structures for cases like (17)b, which has two readings depending on what *mesci-un* modifies.[5]

(17) a. [John-uy [mesci-in os]]
John-Gen fancy-Rel clothes
'John's fancy clothes'

b. mesci-un ku sinsa-uy os
fancy-Rel that gentleman-Gen clothes
'the clothes of the good-looking gentleman's or the fancy clothes of the gentleman's'

3.6 Topicalization

The grammar also generates both simple and complex topic clauses:

(18) a. John-i hangsang ku chayk-ul ilk-ess-ta
John-NOM always that book-ACC read-PST-DECL
'John always read that book.'

b. ku chayk-un [John-i hangsang __ ilk-ess-ta]
that book-TOP John-NOM always read-PST-DECL
'As for the book, John always read it.'

Korean displays various topicalization cases where one element of the given sentence is topicalized to the sentence initial position as in (18)b. The well-formed phrase condition responsible for generating such cases is *head-filler-ph*:

[5]The approach we adopts for relative clauses are adopted to parse topicalization sentences in a similar manner.

```
hd-filler-ph := phrase & binary &
 [ SYN.HEAD verb,
   SYN.VAL [ ARG-ST <>,
             GAP <! !> ],
   ARGS < phrase & #1 & [ SYN.VAL.ARG-ST <> ],
          phrase & [ SYN.VAL.ARG-ST <>,
                     SYN.VAL.GAP <! #1 !> ] > ].
```

The condition here simply says that a head element with a gap can combine with the filler whose syntactic and semantic information is structure sharing with that of the gap. This induces a simple analysis for canonical as well as long-distance topicalized cases such as (19):

(19) ku chayk-un [John-i [Mary-ka hangsang __ ilk-ess-ta-ko]]
that book-TOP John-NOM Mary-NOM always read-PST-DECL-DECL

mit-nun-ta
believe-PRES-DECL
'As for the book, John believes Mary always read it.'

The information that there exists one gapped element in the embedded clause is passed up to the point where the gap meets the filler information.[6]

3.7 Complex Predicates

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 somehow or other. The main constructions the grammar covers at this stage are auxiliary and light verb constructions. 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 (b) the V2 takes an agentive subject but inherits its other arguments to the final predicate V2, and (c) V1 and V2 form a tight syntactic unit.

3.8 Auxiliary Constructions

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 (20)a. 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 (20)b:

(20) a. John-un sakwa-lul mek-ko/*e siph-ess-ta.
John-Top apple-Acc eat-Comp/Comp like-Pst-Decl
'John wanted to eat apples.'

b. *sakwa-lul John-un mek-ko 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 (21).

(21) a. John-i pyonci-lul hyucithong-ey neh-e peli-ess-ta
John-Nom letter-Acc garbage.can-LOC put-COMP do-Pst-Decl
'John has put the letter into the garbage can.'

[6]One tricky issue here is that the topic-marked phrase can be used as a contrastive element. The system captures this fact with respect to case and discourse markers.

b. John-un wul-e peli-ess-ta
John 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 (22) is added in process of forming the complex predicate.

(22) 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 grammar developed here attributes such basic properties to lexical information of auxiliary verbs, the constraints on the phrase *head-word-ph*, and the mechanism of argument composition (Bratt 1996, Kim 2000). The following is an illustration for the type definition of auxiliary verbs and an argument composition rule:

```
v-aux-v := aux-v &
 [ SYN.VAL.ARG-ST.FIRST phrase & [ SYN.HEAD [ CASE nom, PRD - ],
                                   SEM.INDEX #arg1 ],
   SEM.KEY.ARG1 #arg1 ].

head-wd-arg-rule-1 := hd-wd-ph &
 [ SYN.VAL.ARG-ST #argst,
   ARGS < word & #2 & [ SYN.VAL.ARG-ST #argst & [ FIRST #1 ] ],
          v-word & [ SYN.VAL.ARG-ST < #1, #2 > ] > ].
```

The rule here specifies a licit case of *hd-wd-ph*: the auxiliary verb, being the second element in the ARGS, selects two arguments the subject (#1) and the main verb (#2) which in turn selects at least the identical subject. In such a case, the main verb's ARG-ST value will be identical to the resulting phrase.

In addition to the successful parsing of various related constructions, the analysis brings us other desirable consequences. For example, such an analysis can be directly adopted to the treatment of light-verb constructions. The light verb constructions share various properties with auxiliary constructions. The only difference is the lexical argument of the light verb is a verbal noun:

(23) Ku hoysa-un Mikwuk-ey catongcha-lul swuchwul ha-ess-ta
the company-Top US-Dat car-Acc export do-Past-Decl
'The company exported cars to the United States.'

The present grammar needs no additional mechanism in parsing such constructions, other than the lexical information of the light verb *ha-* as follows:

```
light-v := pos-main-v &
 [ SYN.VAL.ARG-ST < phrase & [ SYN.HEAD.CASE nom,
                               SEM.INDEX #arg1 ],
                    n-word & [ SYN.HEAD.VERBAL +,
                               SEM.INDEX #arg2 ] >,
   SEM.KEY [ ARG1 #arg1,
             ARG2 #arg2 ] ].
```

The parsing results of various complex predicate constructions with phenomena such as negation and relative clause further prove the validity and efficiency of the grammar.

4 Concluding Remarks

We thus have first developed a Korean Phrase Structure Grammar couched in the constraint-based framework of HPSG and then checked its feasibility through the implementations into the LKB system.

The work described here, even though it is an on-going project, achieves impressive coverage of major constructions in the language in question, providing a promising future direction. The research presented in this paper provides promising 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.

References

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.

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

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

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

Kim, Jong-Bok. 1998a. A Head-driven and Constraint-Based Analysis of Korean Relative Clause Constructions: With a Reference to English. *Language Research 34.4*: 1–41.

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

Sag, Ivan and Tom Wasow. 1999. *Syntactic Theory: A Formal Approach.* Stanford: CSLI Publications.

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.