A Linear Computational System for Korean: Case and Structure*

William O'Grady
(University of Hawaii)

1. Introduction

This paper seeks to extend to Korean and other SOV languages an idea that I have developed in some detail for SVO languages such as English (O'Grady, to appear). Put simply, that idea is that there is no grammar. Rather, sentences are formed and interpreted by an efficiency-driven processor whose operation is designed to minimize the burden on working memory.

As a preliminary illustration of how this might work, let us consider the formation of the sentence *Mary drank water*. The starting point of course lies in the lexical properties of the verb *drink*. As illustrated below, the lexical entry for this verb indicates that it has two argument dependencies; in particular, it requires two nominal arguments, one to its left and the other to its right.

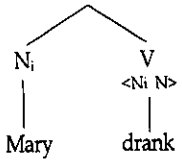
(1) drink: V, <N N>
<— —>

Argument dependencies are resolved by a computational system that is indistinguishable from a processor. This system operates from left to right,

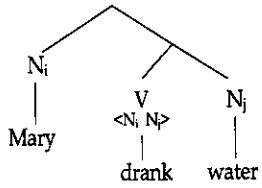
* I thank Miho Choo for her comments and editorial assistance.

combining the verb with its arguments one at a time at the first opportunity. (In the example that follows, the resolution of argument dependencies is represented by copying the index of the argument into the appropriate slot in the verb's grid.)

(2) a. First step: combination of the verb with the nominal to its left; resolution of its first argument dependency



b. Second step: combination of the verb with the nominal to its right; resolution of its second argument dependency



In a system such as this, the core properties of syntactic phenomena follow not from autonomous grammatical principles, but simply from the manner in which sentences are built. In the case at hand, for instance, we end up with a binary-branching structure in which the subject is higher than the direct object. Crucially, though, this does not come about because of a grammatical stipulation (such as the X-bar schema). Rather the sentence's design follows from the manner in which it is built—left to right, one word at a time, with each dependency resolved at the first opportunity.

As explained in O'Grady (to appear), the properties of a large number of 'core' syntactic phenomena—binding, control, agreement, extraction, contraction,

and so forth—can be accounted for in this way. The purpose of this paper is investigate the prospects of a similar approach to various problems that arise in the syntax of SOV languages, especially Korean.

There are clearly major challenges here, since the clause-final position of verbs raises difficult questions about how and whether structure can be built in a left-to-right fashion in SOV languages. I believe not only that it can, but that proceeding in this way sheds light on a number of important phenomena in the syntax of Korean, including the precise role of case.

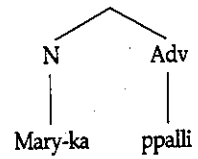
2. Building structure in Korean

How then might the sort of computational system sketched above build a sentence such as the following?

(3) Mary-ka ppalli ttena-ss-ta.
Mary-Nom quickly leave-Pst-Decl
'Mary quickly left.'

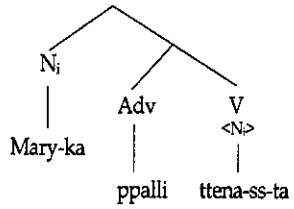
My proposal is that the computational system proceeds one step at a time, combining each word with a preceding element. Thus in the first step, the nominal *Mary* is combined with the adverb *ppalli* 'quickly', yielding the intermediate representation in (4).

(4)



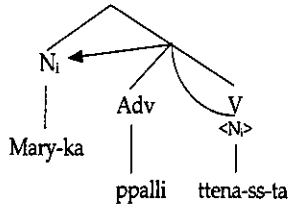
No dependencies can be resolved here, but this quickly changes with the addition of the verb, as illustrated below.

(5)



Here, combination of *ttena-ss-ta* 'left' with *ppalli* 'quickly' satisfies the adverb's need for a verb. The verb's dependency on a nominal argument is then passed upward ('feature passing') for resolution by the nominal *Mary*.

(6)



There are at least two advantages of proceeding in this way: First, by doing so, the computational system builds the sort of hierarchically structured binary representations that are eventually necessary anyhow. That is, if allowed to run its course, the operation that yields the temporary constituent *Mary-ka ppalli* will eventually yield the desired binary-branching constituent structure for the entire sentence depicted in (5) above.

A second, and perhaps more fundamental advantage of this way of sentence building is that it minimizes the burden on working memory by immediately structuring the input. In fact, it is a virtual article of faith in the literature on sentence processing that the input should be organized as quickly as possible, as evidenced for example by Frazier's (1987: 583) suggestion that the processor

adopts as its overarching strategy a simple principle: 'Structure the input as soon as possible'.

So far, so good. But we must still ask whether there is evidence that things really do work this way. I believe that there is. A sample piece of support comes from phonology.

Phonological evidence

A key assumption underlying my view of phonology is that phonological operations take place in real time, as the words making up a sentence combine with each other. With that in mind, consider the process that nasalizes an obstruent in front of a nasal within the same intonational phrase (e.g., Jun 1993: 120).

(7) obstruent → +nasal / _ +nasal

The prototypical examples of this phenomenon take place within compounds.

(8) a. [m] b. [ng]
aph mwun cak nyen
front door last year

Intuitively, what we want to say here is that nasalization applies at the point where a word ending in an obstruent combines with a word beginning with a nasal.

But now consider the following examples.

(9) a. Subject + verb
[ng]
Hopak manh-ta.
pumpkin be.many-Decl
'Pumpkins are numerous.'

b. subject + adverb + verb

[ng]

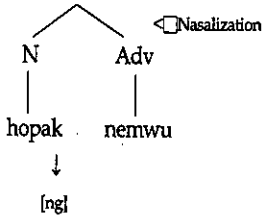
Hopak nemwu manh-ta.

pumpkin overly be.many-Decl

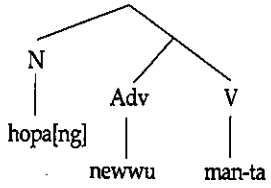
'Pumpkins are overly numerous.'

Nasalization in the first case is unremarkable, since there is no doubt that the subject nominal *hopak* 'pumpkin' combines with the predicate *manh-ta* 'be many'. However, the second case is more interesting since nasalization occurs when the subject nominal makes contact with the adjacent adverb. The proposed system of sentence formation allows us to capture this straightforwardly: nasalization takes place at the point where the computational system combines the subject nominal with the adverb, creating the temporary constituent *hopak nemwu* 'pumpkins overly', which is subsequently restructured by combination of the verb with the adverb.

(10) a. Combination of the nominal and the adverb



b. Combination of the adverb and the verb



A good deal of other evidence—phonological, morphological, and syntactic—points toward the same conclusion: words are combined more or less as they are encountered, even if the resulting phrase is not permanent. Space does not permit a survey of this evidence here, however, and I will turn instead to my central point, which involves the status and role of case in a sentence-building system of this sort.

3. The role of case

The basic intuition that I have worked with for many years (see, e.g., O'Grady 1991) is that case carries information about the type of category with which a nominal combines. In particular:

(11) a. The accusative (*-ul/lul*) records combination with a transitive verbal category (TV).[1]

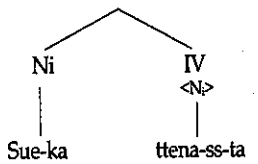
b. The nominative (*-i/ka*) records combination with a non-transitive verbal category (IV).

c. The genitive (*-uy*) records combination with a nominal.

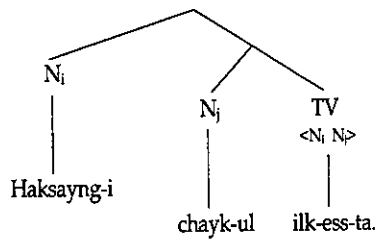
I assume that, by definition, a transitive verbal category depends on two nominal arguments. All other verbal categories are non-transitive.

Focusing for a moment just on representations, rather than on how they are built, we can see how case works by considering the following two examples.

(12)



(13)



1 This is an approximation, I admit. The class of verbs with which an accusative-marked nominal combines includes, but is not restricted to, transitive verbs.

The nominative case in (12) marks a nominal that combines with an intransitive verb, while the accusative case in (13) appears on a nominal that combines with a transitive verb, just as we would expect. But what of the *haksayng*, which also carries the nominative case? It too combines with a category of the right type since the verbal phrase *chayk-ul ilk-ess-ta* 'read the book' is not transitive—it is looking for a single nominal argument. (Of course, ilk-ta 'read' by itself is transitive.)

In later work (e.g., O'Grady 2002, 2003), I proposed that case markers are themselves functors (i.e., argument-taking categories) that add to a nominal a dependency on a verbal category of the appropriate type. Thus, a nominative-marked nominal exhibits a dependency on a non-transitive verbal category, while an accusative-marked nominal carries a dependency on a transitive verbal category.

(14) a. N <IV> | Sue-ka b. N <TV> | chayk-ul

These dependencies are then resolved by combination with a category of the appropriate type—a non-transitive verb in the case of the nominative and a transitive verb in the case of the accusative. (I use a check mark to indicate that a case dependency has been resolved.)

(15) a. Ni <IV√> — Sue-ka; IV <Ni> — ttena-ss- b. Nj <TV√> — chayk-ul; TV <N Nj> — ilk-ess-ta

As depicted here, combination not only resolves the relevant 'case dependencies', it also leads to the resolution of the verb's argument dependencies. Hence, as indicated by the indexing, Sue is interpreted as the subject argument of *ttena-ss-ta*, and *chayk-ul* is identified as the object argument of *ilk-ess-ta*.

3.1 How case directs the computational system

As the preceding examples help illustrate, case guides the operation of the computational system by ensuring that particular nominals are combined with particular functors. Let us now consider how this might work in an efficiency-driven linear computational system of the sort considered in section 2. The key idea is as follows.[2]

(16) The regulatory role of case:
A nominal is not eligible to resolve an argument dependency until its case dependency has been resolved.

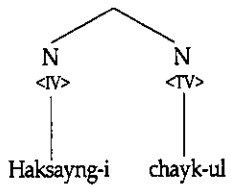
The formation of the simple sentence in (17) illustrates this in a preliminary way.

(17) Haksayng-i chayk-ul ilk-ess-ta.
student-Nombook-Acc read.Pst.Decl
'The student read a book.'

Proceeding from left to right, the computational system begins by combining the nominative-marked nominal and the accusative-marked nominal. No dependencies are resolved at this point.

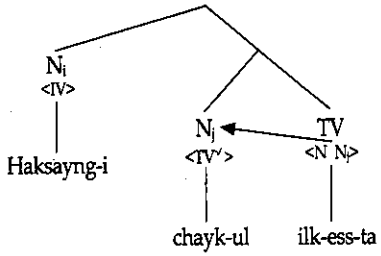
2 I take this to be a processing constraint whose function is to minimize garden paths.

(18)



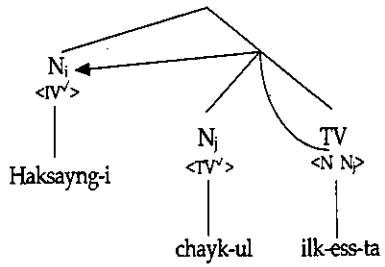
The next step involves combination of the verb with the accusative-marked nominal. This leads to resolution of the dependency associated with the accusative case, followed by resolution of the verb's theme argument dependency.

(19)



The dependency associated with the nominative case can also be resolved, followed by resolution of the verb's agent argument dependency with the help of feature passing. (Recall that the verbal phrase *chayk-ul ilk-ess-ta* 'read the book' is not transitive since it exhibits a dependency on a single nominal argument; see the discussion of (13) above.)

(20)



3.2 A case contrast involving causatives

Now let us consider the more challenging problem presented by the familiar case alternation illustrated in the following pair of causative sentences.

(21) a. The nominative-nominative pattern

Mary-**ka**	John-**i**	ttena-key	hay-ss-ta.
Mary-**Nom**	John-**Nom**	leave-Comp	do-Pst-Decl

'Mary made John leave.'

b. The nominative-accusative pattern

Mary-**ka**	John-**ul**	ttena-key	hay-ss-ta.
Mary-**Nom**	John-**Acc**	leave-Comp	do-Pst-Decl

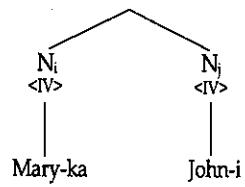
'Mary made John leave.'

It is well known that these two patterns differ both syntactically and semantically. If we are on the right track, these differences should follow from the manner in which the two sentences are built, as directed by the case marking. Let us consider each pattern in turn.

The nominative-nominative pattern

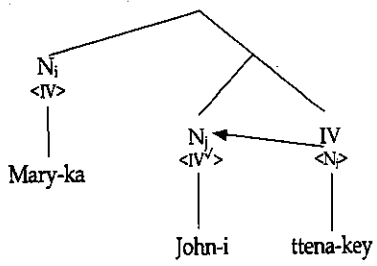
The first step in the formation of the double nominative pattern involves combination of the nominals *Mary* and *John*. Of course, no dependencies can be resolved at this point.

(22)



In the next step, the computational system combines the nominative-marked nominal John and the intransitive verb *ttena-key*, thereby resolving the case dependency and permitting resolution of the verb's argument dependency. (Recall that a nominal is not eligible to resolve an argument dependency until its case dependency has been resolved.)

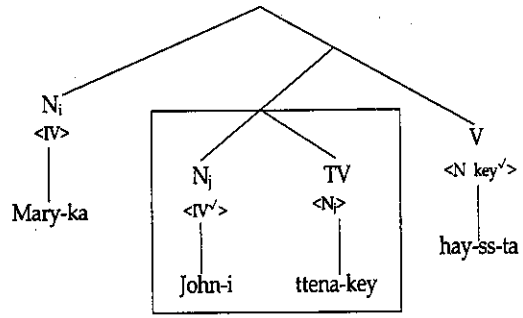
(23)



Because John-i *ttena-key* exhibits no unresolved dependencies, let us assume that it is 'frozen' and henceforth behaves like an indivisible unit rather than two separable words.

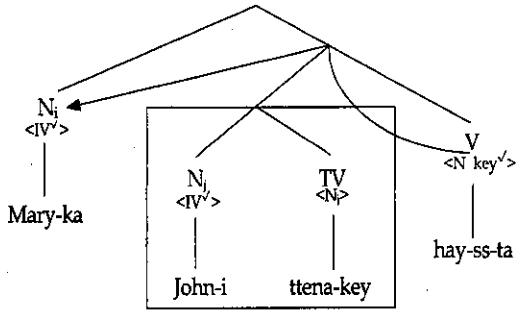
Next comes addition of the causative verb *ha-ta*. (For the sake of exposition, I assume only that *ha-ta* takes a 'key-phrase' as its complement, without taking a position on the category of this phrase.)

(24)



Because the resulting phrase (*John-i ttena-key hay-ss-ta* 'made John leave') is not transitive, the dependency associated with the nominative case on *Mary* can also be resolved. This in turn permits *Mary* to resolve *ha-ta*'s remaining argument dependency (with the help of feature passing).

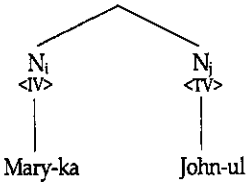
(25)



The nominative-accusative pattern

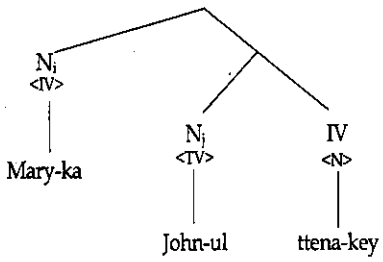
Things turn out somewhat differently in the case of the nominative- accusative pattern. In the first step, the computational system once again combines the nominals *Mary* and *John*, even though no dependencies can be resolved at this point.

(26)



Next comes combination of *John* and *ttena-key* 'leave'.

(27)

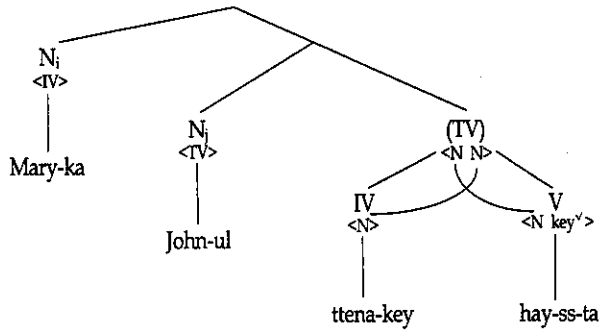


No dependencies can be resolved at this point either, since the nominal carries an accusative case marker, signaling that it is looking for a TRANSITIVE verb. (Remember once again that a nominal cannot be used to resolve an argument dependency until its case dependency has been resolved.)

The time has now come to add the causative verb *ha-ta*, but what should it combine with? In contrast with the situation in the nominative-nominative pattern, *John(-ul) ttena-key* is not a possible target since its component parts have not fused. That was prevented by the accusative case marker: because the case dependency cannot be resolved, *John* is not able to serve as argument of *ttena-ta*.

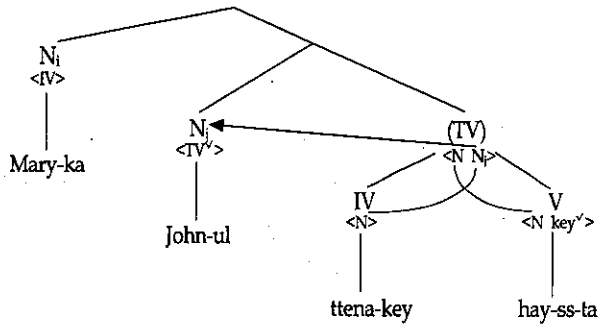
There is just one possibility, then. The computational system must combine *ha-ta* with just *ttena-key*. The result (*ttena-key hay-ss-ta* 'caused to leave') is a transitive verbal phrase thanks to the presence of two nominal dependencies, one corresponding to the sole argument of *ttena-key* and the other corresponding to the subject argument of *ha-ta*. This is illustrated below.

(28)



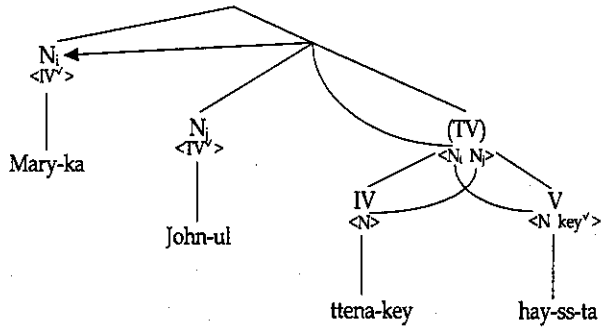
This result is exactly right, since it leads to resolution of the dependency associated with the accusative case on *John*, which can then be used to resolve one of the verb's argument dependencies.

(29)



In addition, the computational system is able to resolve the dependency associated with the nominative case on *Mary*, which in turn permits the use of this nominal to resolve the verb's remaining argument dependency.

(30)



Some independent evidence

As can be seen here, case is very much 'in charge'—it essentially drives the computational system, telling it at what point it can resolve dependencies and

thereby determining the course of the sentence formation process. A particularly interesting feature of this perspective on case and structure building is that it yields representations with exactly the properties that have been independently attested. I will mention just two examples from the literature on this subject (see, e.g., O'Grady 1991, Sells & Cho 1991, and Bratt 1993).

Let us begin with the idea put forward by Haiman (1983) according to which more direct syntactic relationships express more direct semantic relationships. Because the causative verb *ha-ta* is forced to combine directly with the embedded verb in the nominative-accusative pattern, we would expect that construction to denote a relatively direct sort of causation. In contrast, we would expect a less direct sort of causation to be associated with the nominative-nominative pattern, in which the causative verb combines with the entire embedded clause. This seems to be right: it is widely reported that the nominative-accusative pattern expresses a more direct type of causation than does the nominative-nominative pattern (e.g., O'Grady 1991: 189 and the references cited there).

A quite different sort of phenomenon involves the scope of the manner adverb in patterns such as (31).

(31) a. The nominative-nominative pattern
Mary-ka [John-i kuphi ttena-key] hay-ss-ta.
Mary-Nom John-Nom quickly leave-Comp make-Pst-Decl
'Mary made John leave quickly.'
b. The nominative-accusative pattern
Mary-ka John-**ul** kuphi ttena-key hay-ss-ta.
Mary-Nom John-Acc quickly leave-Comp make-Pst-Decl
'Mary made John leave quickly.'
or: 'Mary quickly made John leave.'

Because the adverb occurs in the middle of the embedded clause in the nominative-nominative pattern, its scope is restricted to the verb in that clause.

Matters are different in the case of the nominative-accusative pattern, where the case marking effectively blocks the formation of an embedded clause, leaving the adverb free to modify either just *ttena-key* or the larger verbal complex headed by *ha-ta*.

4. Conclusion

The standard view, both in the traditional descriptive literature and in much of the theoretical literature (see Kim 2004 for a recent example), is that the primary function of case is to signal grammatical relations such as subject and direct object. If the ideas put forward here are on the right track, though, it may be necessary to rethink this assumption. As I see it, the function of case is to regulate and record the operation of the processor by determining the point at which dependencies are resolved and (permanent) phrases are formed.

An especially clear example of this comes from the familiar contrast between the two causative patterns that we have been considering. In the nominative-nominative construction exemplified by (31a), the case marker permits the embedded verb to immediately resolve its argument dependency upon combination with the nominal *John*, thereby forming a permanent phrase. In contrast, a different case marker on this same nominal in the nominative-accusative pattern prevents this from happening, ensuring that the sentence will have a very different structure, with the consequences we have observed.

This view of case is of course embedded within a larger framework of assumptions about the nature of the computational system for language that has yet to be proven tenable. Nonetheless, early results suggest that further inquiry along these lines may be worthwhile, for both empirical and conceptual reasons. In any event, it seems clear that work on the syntax of Korean will be vital for determining the ultimate viability of this approach to structure and case.

References

Bratt, Elizabeth. 1993. Clause structure and case marking in Korean causatives. Harvard Studies in Korean Linguistics V, 241-51.

Frazier, Lyn. 1987. Sentence processing: A tutorial review. In M. Coltheart (ed.), Attention and performance XII: The psychology of reading. Hillsdale, NJ: Erlbaum. Pp. 559-86.

Haiman, John. 1983. Iconic and economic motivation. Language 59, 781-819.

Jun, Sun-Ah. 1993. The phonetics and phonology of Korean prosody. Ph.D. dissertation. University of California at Los Angeles.

Kim, Jong-Bok. 2004. Korean case system: A unified, constraint-based approach. This volume.

O'Grady, William. 1991. Categories and case: The sentence structure of Korean. Philadelphia: John Benjamins.

———. 2002. Korean case: Extending the computational approach. Korean Linguistics 11, 29-51.

———. 2003. A computational approach to case and word order in Korean. In Y. A. Lee & A. Simpson (eds.), Functional structure(s), form and interpretation: Perspectives from east Asian languages. New York: Routledge-Curzo. Pp. 222-40.

O'Grady, William. to appear. Syntactic carpentry: An emergentist approach to syntax. Mahwah, NJ: Erlbaum.

Sells, Peter & Young-mee Yu Cho. 1991. On the distribution of X° elements in Korean. Proceedings of the Santa Cruz Workshop on Korean Syntax and Semantics.

A Computational Treatment of Some Case-Related Problems in Korean

Kiyong Lee*
(Korea University)

1. Preamble: Aim and Basic Assumptions

This work has two related aims: one is general and long-range and the other, specific and of immediate concern. The long-range, general aim is the construction of a simple or minimal syntax that will generate semantic representations in feature structure. It is implemented in a C-augmented language called MALAGA.[1] The immediate goal is to show the computational tractability of some case-related problems in Korean by constructing a MALAGA-based syntactic system named KorSyn.[2]

* Emeritus, Korea University, klee@korea.ac.kr.

** I owe many thanks to Suk-Jin Chang and Jong-Bok Kim and many others of the Korean HPSG Working Group members whose comments I benefitted very much for improving this paper, and also to Jae-Woong Choe for his proofreading and comments, Roland Hausser for his encouraging comments, and Tyrone Cashman who read and made editorial and stylistic comments on this paper during his short visit to Korea. Jungha Hong helped me to expand some parts of the program and fix some bugs in it.

1 MALAGA stands for 'Malaga Accommodates Left-Associative Grammar with Attribute-value data structures'. Developed by Beutel (2003) and his colleagues at Department of Computational Linguistics, University of Erlangen-Nuernberg under Prof. Roland Hausser, it is a C-type implementation language augmented for both programming and debugging purposes and particularly made suitable for implementing various Left-Associative Grammars of natural human language.

2 Currently, MALAGA is being replaced by a Java-based grammar development tool named Java LAG that makes it more efficient to implement Database Semantics. See Lee (1999b,