

from *Proceedings of the 2004 Linguistic Society of Korea International Conference, Volume I: Forum Lectures and Workshops*. Seoul: Linguistic Society of Korea, 2004, 182-94.

Rethinking Structure and Case*

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 2005). 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.

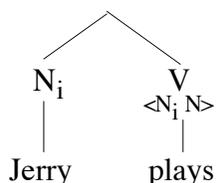
As a preliminary illustration of how this might work, let us consider the formation of the sentence *Jerry plays chess*. The starting point of course lies in the lexical properties of the verb *play*. As illustrated below, the lexical entry for this verb indicates that it has two argument dependencies—in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

(1) *play*: V, <N N>
 <- ->

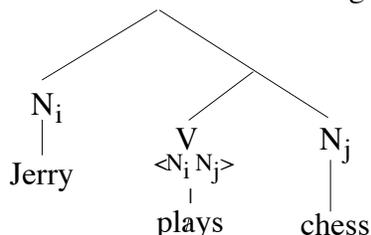
Argument dependencies are resolved by a computational system that is indistinguishable from a processor. This system operates from left to right, combining the verb with its arguments one at a time at the first opportunity. (In the example that follows, the resolution of argument dependencies is represented by copying the index of the argument into the appropriate slot in the verb's grid.)

*Although this paper deals with a different case phenomenon, certain sections draw heavily on O'Grady (2004), which is meant as a 'companion piece'. I thank Miho Choo for her comments and editorial assistance.

- (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 (2005), 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 to investigate the prospects of a similar approach to various problems that arise in the syntax of SOV languages, especially Korean.

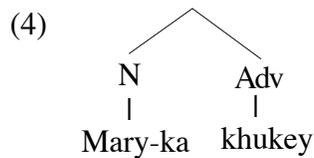
There are clearly major challenges here, since the clause-final position of verbs raises difficult questions about how and whether structure can be built in a left-to-right fashion in SOV languages. I believe not only that it can, but that proceeding in this way sheds light on a number of important phenomena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

2. Building structure in Korean

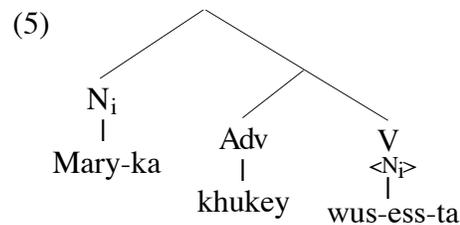
How then might the sort of computational system sketched above build a sentence such as the following?

- (3) Mary-ka khu-key wus-ess-ta.
 Mary-Nom loudly laugh-Pst-Decl
 ‘Mary laughed loudly.’

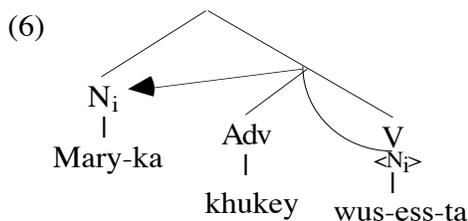
My proposal is that the computational system proceeds one step at a time, combining each word with a preceding element. Thus in the first step, the nominal *Mary* is combined with the adverb *khukey* ‘loudly’, yielding the intermediate representation in (4).



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



Here, combination of *wus-ess-ta* ‘laughed’ with *khukey* ‘loudly’ 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*.



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 khukey* 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, there is virtual consensus 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. On this view, assimilatory processes are in a sense iconic, reflecting the phonological merger that accompanies and signals syntactic combination.

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, Choo & O'Grady 2003:77-78).

(7) obstruent \rightarrow +nasal / _ +nasal

The prototypical examples of this phenomenon take place within compounds.

| | | | |
|--------|-----------------|----|------------------|
| (8) a. | [m] | b. | [ng] |
| | aph mwun | | ca k 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

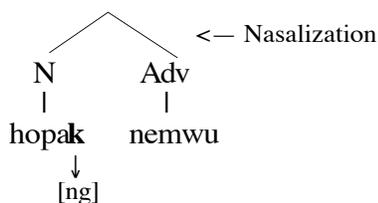
[ng]
Hopak^k manh-ta.
pumpkin be.many-Decl
'Pumpkins are numerous.'

b. subject + adverb + verb

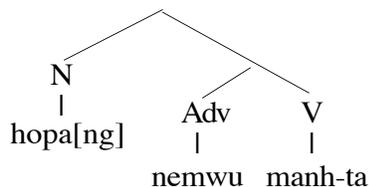
[ng]
Hopak^k nemwu manh-ta.
pumpkin overly be.many-Decl
'Pumpkins are overly numerous.'

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.

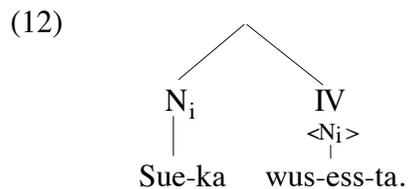
3. The role of case

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:

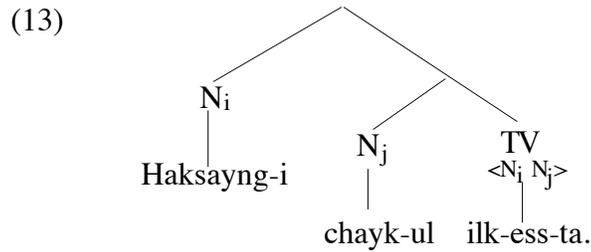
- (11)a. The accusative (*-ul/lul*) records combination with a transitive verbal category (TV).¹
 b. The nominative (*-i/ka*) records combination with a non-transitive verbal category (IV).
 c. The genitive (*-uy*) records combination with a nominal.

I assume that, by definition, a transitive verbal category manifests two nominal argument dependencies, one of which is agentive. 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.



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

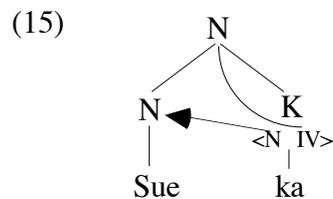


The nominative case in (12) marks a nominal that combines with an intransitive verb, while the accusative case in (13) appears on a nominal that combines with a transitive verb, just as we would expect. But what of the *haksayng*, which also carries the nominative case? It too combines with a category 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 seek out a nominal argument (the noun stem) and a second category of the appropriate type—a non-transitive verbal category in the case of the nominative, a transitive verbal category in the case of the accusative, and another nominal in the case of the genitive.

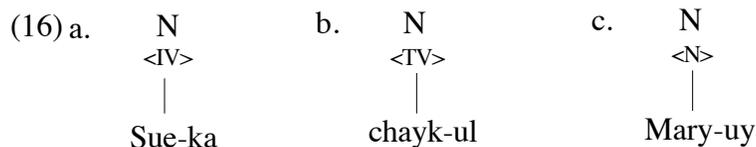
- (14)a. *-i/ka* (nominative case): <N IV>
 b. *-ul/lul* (accusative case): <N TV>
 c. *-uy* (genitive case): <N N>

Combination of the case marker with the noun stem resolves the first of these dependencies. The remaining dependency is inherited (via feature passing) by the resulting nominal phrase. For example: (K = case)

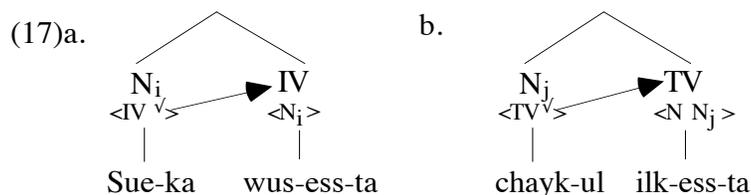


Thus, a nominative-marked nominal exhibits a dependency on a non-transitive verbal category, while an accusative-marked nominal carries a

dependency on a transitive verbal category and a genitive-marked nominal manifests a dependency on a nominal category. We can capture these facts with the help of the following simplified notation, bypassing the internal structure of nominals, which is not relevant here.



These dependencies are 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, for example. (I use a check mark to indicate that a case dependency has been resolved.)



In these examples, combination of the verb and the nominal not only resolves the relevant ‘case dependencies’, it also leads to the resolution of an argument dependency. 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.

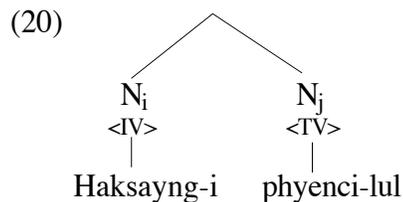
- (18) The regulatory role of case:
 A nominal is not eligible to resolve an argument dependency until its case dependency has been resolved.

I take this to be a processing constraint that is designed to minimize garden paths by preventing false linking of functors and arguments.

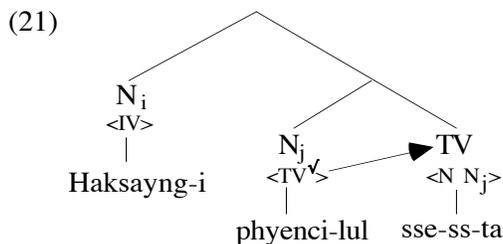
The formation of the simple sentence in (19) illustrates this in a preliminary way.

- (19) Haksayng-i phyenci-lul sse-ss-ta.
 student-Nom letter-Acc write-Pst-Decl
 ‘The student wrote a letter.’

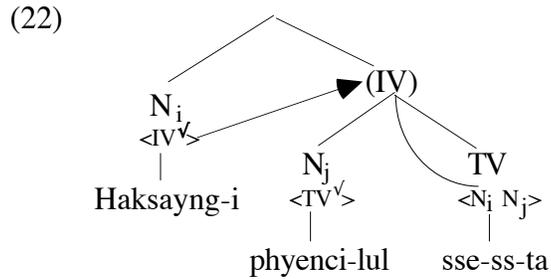
Proceeding from left to right, the computational system begins by combining the nominative-marked nominal and the accusative-marked nominal. No dependencies are resolved at this point.



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.



At this point, the dependency associated with the nominative case can be resolved, followed by resolution of the verb’s agent argument dependency with the help of feature passing. (Recall that the verbal phrase *phyenci-lul sse-ss-ta* ‘wrote the letter’ is not transitive since it exhibits a dependency on a single nominal argument; see the discussion of (13) above.)



3.2 The genitive-nominative alternation

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

- (23)a. Genitive pattern:
 Mary-**uy** elkwul-**i** yeppu-ta.
 Mary-Gen face-Nom pretty-Decl
 ‘Mary’s face is pretty.’
- b. Double nominative pattern:
 Mary-**ka** elkwul-**i** yeppu-ta.
 Mary-Nom face-Nom pretty-Decl
 ‘Mary’s face is pretty.’

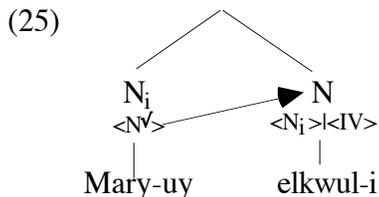
It is well known that these two patterns differ both syntactically and semantically. If we are on the right track, these differences should follow from the manner in which the two sentences are built, as directed by the case marking. Let us consider each pattern in turn.

The genitive pattern

A key to understanding the formation of both patterns lies in the assumption that ‘body part’ nominals such as *elkwul* ‘face’, *son* ‘hand’, *nwun* ‘eye’, and so forth are functors that take a possessor argument. Hence the lexical entry for *elkwul* resembles (24).

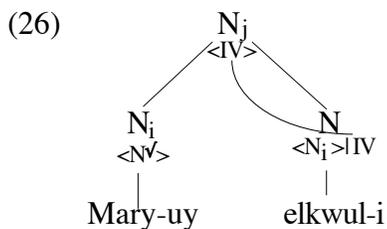
- (24) *elkwul*: N
 <N>
 poss

The first step in the formation of the genitive pattern involves combination of the nominals *Mary* and *elkwul*. (I use a vertical bar to separate the argument dependency associated with *elkwul* from the case dependency associated with its nominative suffix.)

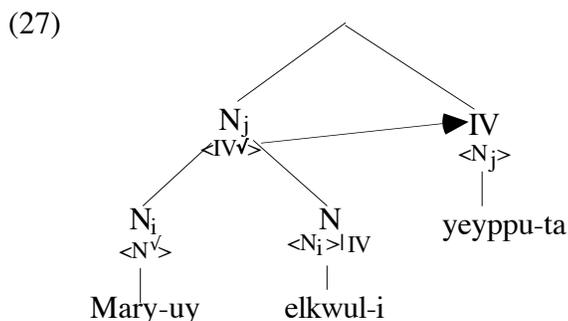


This takes care of the genitive case dependency and permits the nominal *Mary* to resolve the possessor argument dependency associated with *elkwul*. (Recall that a nominal is not eligible to resolve an argument dependency until its case dependency has been resolved.)

The resolution of the argument dependency that takes place here has a consequence of its own: it results in the semantic fusion of *Mary-uy* and *elkwul-i*, creating a phrase that henceforth behaves like an indivisible unit rather than two separable words. In addition, the phrase inherits the unresolved case dependency associated with the nominative suffix on *elkwul*, ensuring that it will eventually have to combine with an intransitive verb.

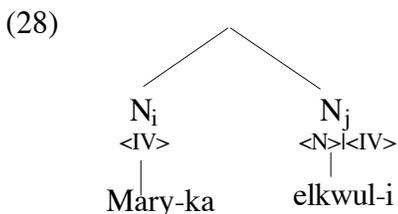


In fact, this is precisely what happens next: the computational system combines the intransitive verb *yeyppu-ta* ‘pretty’ with *Mary-uy elkwul-i*, thereby resolving the nominative case dependency and permitting resolution of the verb’s argument dependency.



The double nominative pattern

Things turn out somewhat differently in the case of the double nominative pattern. In the first step, the computational system combines the nominals *Mary-ka* and *elkwul*.



By proceeding in this way, the computational system is able to immediately structure the input and to take the first step toward building the binary-branching structural representation that will eventually be necessary anyhow. In addition, there is independent phonological evidence that combination takes place at this point.

Recall that I take the view that assimilatory operations are essentially iconic—they reflect the actual COMBINATION of the elements to which they apply, not just their occurrence in adjacent positions. And it turns out that there is at least one assimilatory process that applies at the point where the non-genitive form of *Mary* combines with *elkwul*.

The process in question involves assimilation in place of articulation, which affects morpheme-final /n/ (e.g., Jun 1993, Choo & O'Grady 2003:81). As illustrated in the following examples, a /n/ that occurs in final position of the morpheme X takes on the place of articulation of a consonant at the beginning of the morpheme Y when X and Y combine.

[m]
 (29)a. maywun pap
 spicy rice

[ng]
 b. khun khal
 big knife

Interestingly, we find reflexes of the same process in the topicalized version of the double nominative pattern.

[m]
 (30)a. Na-n pal-i aphu-ta.
 I-Top foot-Nom hurt-Decl
 ‘As for me, (my) feet hurt.’

[ng]
 b. Na-n kho-ka aphu-ta.
 I-Top nose-Nom hurt-Decl
 ‘As for me, (my) nose hurts.’

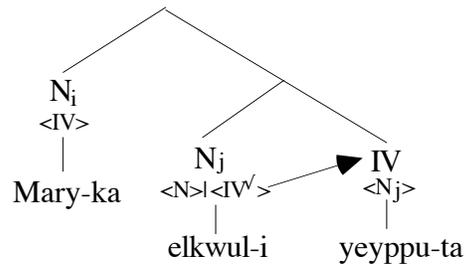
Given the assumptions that we have adopted about the relationship between assimilation and syntactic computation, this strongly suggests that the possessor and the possessee combine even where there is no immediate opportunity to resolve argument dependencies, exactly as depicted in (27).

Crucially, however, no dependencies are resolved at this point—since the nominative case dependency associated with *Mary* cannot be satisfied, that nominal is not eligible to resolve the possessor argument dependency associated with *elkwul*. The computational system must therefore move on to the next word.

The next word is the verb, but what should it combine with? In contrast with the situation in the genitive pattern, *Mary-ka elkwul-i* is not a possible target since its component parts have not fused. That was prevented by the presence of the nominative case marker on the first nominal: because the case dependency cannot be resolved, *Mary* is not able to serve as argument of *elkwul*.

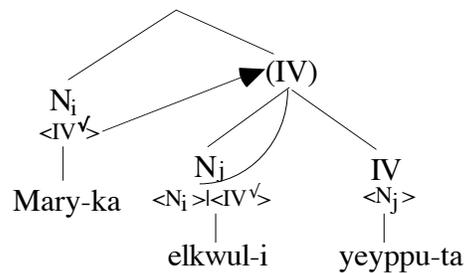
There is just one possibility, then. The computational system must combine the intransitive verb *yeyppu-ta* with just *elkwul-i*. This resolves the nominative case dependency associated with *elkwul-i*, which in turn permits that nominal to resolve the verb's argument dependency.

(31)



As illustrated below, the phrase *elkwul-i yeyppu-ta* is itself intransitive, since it inherits the unresolved possessor argument dependency associated with *elkwul*. This permits resolution of the case dependency associated with *Mary*, which in turn makes it possible for that nominal to resolve the remaining argument dependency. As a result, *Mary* is ultimately identified as the possessor argument of *elkwul* 'face'.

(32)



Some independent evidence

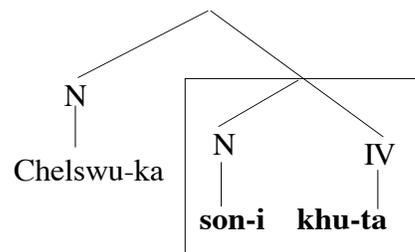
As can be seen here, case is very much ‘in charge’—it essentially drives the computational system, telling it at what point it can resolve dependencies and thereby determining the course of the sentence formation process. A particularly interesting feature of this perspective on case and structure building 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).

A first phenomenon involves the way certain idioms work in Korean. In the simplest case, idioms are basically just pieces of sentences that have taken on a special meaning that is not predictable from the meanings of their parts. Interestingly, Korean has quite a few idioms that include the sort of body part nominals that we have been considering. Examples of this type were first noted by Yoon (1987:145).

- (33) Chelswu-ka son-i khu-ta.
Chelswu-Nom hand-Nom big-Decl
‘Chelswu is a big spender.’ (Lit. ‘Chelswu’s hands are big.’)
- (34) Chelswu-ka pal-i nelp-ta.
Chelswu-Nom foot-Nom wide-Decl
‘Chelswu knows a lot of people.’ (Lit. ‘Chelswu’s feet are wide.’)
- (35) Chelswu-ka kan-i khu-ta.
Chelswu-Nom liver-Nom big-Decl
‘Chelswu is bold.’ (Lit. ‘Chelswu’s liver is big.’)
- (36) Chelswu-ka nwun-i noph-ta.
Chelswu-Nom eye-Nom high-Decl
‘Chelswu has high standards.’ (Lit. ‘Chelswu’s eyes are high.’)

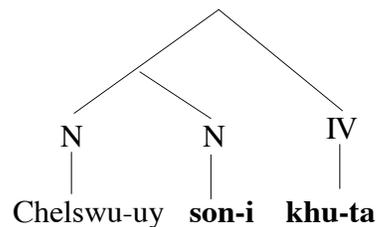
The occurrence of these idioms in double nominative patterns such as these is expected, since the idiomatic portion of the sentence forms a constituent, as illustrated below.

(37)



The same is not true for the genitive pattern, however.

(38)



Very interestingly, the idiomatic interpretation is not so readily available here. For many speakers, the first and only meaning is the literal one: Chelwsu has big hands.

A quite different sort of phenomenon involves the position of time adverbs in patterns such as (39).

(39)a. Genitive pattern:

*Mary-uy nalmata elkwul-i yeypu-ta.

Mary-Gen every.day face-Nom pretty-Decl

‘Mary’s face is pretty every day.’

b. Double nominative pattern:

Mary-ka nalmata elkwul-i yeypu-ta.

Mary-Nom every.day face-Nom pretty-Decl

‘Mary’s face is pretty every day.’

Because adverbs must combine with a verbal projection, the (b) pattern is acceptable. As illustrated in (32) above, *elkwul-i yeypu-ta* constitutes an (intransitive) verbal phrase in the double nominative construction. Matters are very different in the genitive pattern exemplified by (39a), of course.

There, *Mary-uy elkwul* forms a nominal phrase, which naturally rejects an internally placed adverb—just the right result.

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 reveal and regulate 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 genitive-nominative alternation that we have been examining. In the genitive construction, the case marker on the possessor nominal permits it to immediately resolve the argument dependency associated with the body-part nominal, thereby leading to the formation of a permanent nominal phrase (*Mary-uy elkwul* in the example we considered).

In contrast, a nominative marker on the possessor prevents this from happening, ensuring that the sentence will have a very different structure. As we have seen, this difference has important syntactic and semantic consequences.

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

- Choo, Miho & William O'Grady. 2003. *The sounds of Korean*. Honolulu: University of Hawaii Press.

- 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.
- Jun, Sun-Ah. 1993. *The phonetics and phonology of Korean prosody*. Ph.D. dissertation. University of California at Los Angeles.
- Kim, Jong-Bok. 2004. *The Korean case system: A unified, constraint-based approach*. In Byung-Soo Park & Jong-Bok Kim (eds.) *Case phenomena in Korean*. In press.
- 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.
- 2004. *A linear computational system for Korean: Case and structure*. In Byung-Soo Park & Jong-Bok Kim (eds.) *Perspectives on Korean case and case marking*. Seoul: Thaeaksa, Pp. 3-20.
- 2005. *Syntactic carpentry: An emergentist approach to syntax*. Mahwah, NJ: Erlbaum.
- Yoon, James. 1987. *Some queries concerning the syntax of multiple subject constructions in Korean*. *Harvard Studies in Korean Linguistics* 2, 138-62.