IMPLEMENTING SCRAMBLING IN KOREAN: A PRINCIPLES AND PARAMETERS APPROACH

Franklin S. Cho
MIT Artificial Intelligence Laboratory
NE43-802
MIT, Department of Electrical Engineering and Computer Science
545 Technology Square, Cambridge, MA, 02139, USA
fscho@ai.mit.edu

3 July 1995

Summary

This paper describes how the most complete recent linguistic results on Korean scrambling (switching of word order) can be readily incorporated into an existing principles-and-parameters parser with minimal additional machinery. Out of all 29 sets of examples in chapters 2.2 and 3.2 of perhaps the most advanced linguistic analysis of Korean scrambling, (Lee, 1994), 26 sets of examples can be correctly parsed, greatly extending the variety of scrambling handled by any current parser. This approach is compared to other current approaches to scrambling, such as PRINCIPAR and Tree Adjoining Grammar.

Subject Areas: parsing, scrambling, Korean

Word Count:

1. INTRODUCTION

Scrambling is a complex yet common phenomenon in Korean that allows the apparent movement of a noun phrase over both short and long distances:

- Scrambling of more than one noun phrase that belongs to the same verb's argument structure, or "multiple scrambling". For example, in (1)(ii), "chayk" (book) moves in front of "Youlee", or even to the front of the sentence, as in (1)(iv).

(1) (i) Sunhee-ka Youlee-eykey [chayk hankwen]-ul senmwulhayssta
Sunhee-nom Youlee-dat [book one-volume]-acc gave-a-present
"Sunhee gave Youlee a book as a present."

(ii) sunhee-ka [chayk hankwen]-ul youlee-eykey senmwulhayssta
Sunhee-nom [book one-volume]-acc Youlee-dat gave-a-present

(iii) youlee-eykey sunhee-ka [chayk hankwen]-ul senmwulhayssta

(iv) youlee-eykey [chayk hankwen]-ul sunhee-ka senmwulhayssta

(v) [chayk hankwen]-ul sunhee-ka youlee-eykey senmwulhayssta

(vi) [chayk hankwen]-ul youlee-eykey sunhee-ka senmwulhayssta[1]

- No limit to the number of clauses that a scrambled element can cross, or "unbounded dependency," For example, in (2)(ii), "chayk" (book) can be arbitrarily far from its canonical argument position:[2]

(2) (i) John-i [Mary-ka [Sally-ka Bill-eykey chayk-ul cwuessta-ko] malhayssta-ko]
John-nom [Mary-nom [Sally-nom Bill-dat book-acc gave-compl] said-compl]
sayngkakhanta
think
"John thinks that Mary said that Sally gave Bill a book."

(ii) chayk-ul$_i$ [John-i [Mary-ka [Sally-ka Bill-eykey t_i cwuessta-ko] malhayssta-ko]
book-acc [John-nom [Mary-nom [Sally-nom Bill-dat gave-compl] said-compl]
sayngkakhanta]
think]

Handling scrambling correctly is very difficult for a parser. The reason is that adding a permutation component to generate all possible word orders independently of other grammatical constraints is easy. What is more difficult is to make scrambling interact correctly with all the *other* components of the grammar, for instance those that establish the interaction of scrambling with coreference. Consider these examples:

(3) (i) $*$ Younghee-ka **ku**-eykey [**Minswu**-uy sacin]-ul poyecwuessta
Younghee-nom him-dat Minswu-gen picture-acc showed
'Younghee showed **him Minswu's** picture'

(ii) Younghee-ka [**Minswu**-uy sacin]$_i$-ul **ku**-eykey t_i poyecwuessta[3]

(iii) [**Minswu**-uy sacin]$_i$-ul Younghee-ka **ku**-eykey t_i poyecwuessta[4]

The coreference relation is indicated by bold face, and the filler-gap relation (or, equivalently, antecedent-trace relation) by coindexation. (3) shows that scrambling interacts with coreference: the scrambled versions (3)(ii) and (3)(iii) are acceptable, but the canonical version (3)(i) is not. So, scrambling "saves" a sentence in (3). Now, consider these examples:

(4) (i) [**Minswu**-uy tongsayng]-i **ku**-eykey sacin-ul poyecwuessta
Minswu-gen brother-nom him-dat picture-acc showed
'**Minswu's** brother showed **him** a picture.'

(ii) $*$ **ku**-eykey [**Minswu**-uy tongsayng]-i sacin-ul poyecwuessta[5]

Conversely, the canonical version (4)(i) is acceptable but the scrambled version (4)(ii) is not. So, scrambling "destroys" a sentence in (4). Therefore, scrambling is much more complicated than simply generating correct word orders. In both (3) and (4), scrambling interacts with a

[1](Lee, 1994), example 7. See figure 1 in the text.

[2](Lee, 1994), p. 5

[3]Here, "t_i" denotes a link between the canonical argument position for "chayk"(book) and its actual position in the sentence. At this point, I remain theory-neutral as to the exact nature of "t_i". It could be implemented in several ways.

[4](Lee, 1994), example 81.

[5](Lee, 1994), example 82.

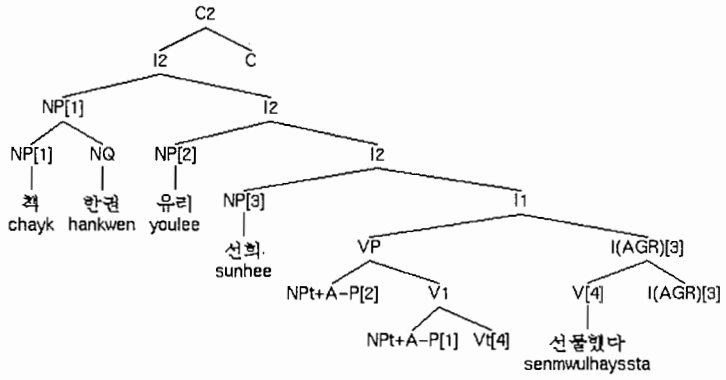

Figure 1: A Parsed Korean Example Sentence 1(vi)

coreference constraint referred to as "Condition C" in the linguistic literature. Roughly, Condition C states that a referring expression e.g., "John," "the house," must not be bound anywhere in a sentence.[6] (4)(ii) is ruled out by Condition C, since "Minswu," a referring expression, is bound by "ku." Such an interaction must be taken into account when parsing these examples. As far as it is known, the system presented here is the first that can correctly parse such a wide range of scrambling examples. Although no good quantitative measures are known to us, scrambling seems quite common in Korean, and is therefore important for parsing. Figure 1 shows an example of the parser's actual output on a scrambled example sentence.

2. IMPLEMENTATION

2.1. A Simple Scrambling Mechanism

Let us see how to parse scrambled sentences using a constraint-based parser, **Pappi** (Fong, 1991). As a first approximation, here is a simplified description of the scrambling mechanism (Figure 2), showing only the modules relevant to the scrambling-coreference relations. In subsequent sections, I will refine this analysis to accomodate new examples. (There are many more filters and generators in **Pappi** than the figure shows.)

The key idea behind constraint-based parsing is to reproduce complex surface sentence patterns by the interaction of separable but linked "modules," each handling a different kind of constraint. For instance, our examples (1)-(4) motivate four modules:

1. One to move NP's from their canonical locations.
2. One to coindex filler NP's with their gaps and other NP's.
3. One to check whether this indexing meets Condition C.
4. One to move variables into proper scope (assuming a typed first-order predicate calculus (FOPC) representation.)

[6] A node A binds another node B iff A and B are coindexed, and A c-commands B. A c-commands B iff the lowest branching node which dominates A also dominates B. For example, in $[\alpha\ [...[\beta\ \theta]...]]$, α c-commands β and θ. This is the canonical definition of binding, and this definition will be modified later, as shown in Figure 4.

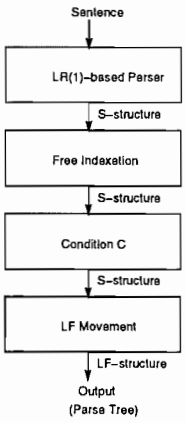

Figure 2: A Simple Scrambling Mechanism

First, a LR(1)-based bottom-up shift-reduce parser is used to recover the phrase structure. Note that this parser allows an NP to freely attach to the beginning of a sentence (or, equivalently, adjoin to an Inflection Phrase(IP)) or to the beginning of a verb phrase (or, equivalently, adjoin to a verb phrase(VP)), to account for scrambling.[7]

A mechanism such as the "Free Indexation" mechanism is called a "generator." A generator generates new structures based on the old structure that was passed to it. For example, the "Free Indexation" generator takes a parse tree without indices, and generates parse trees with indices assigned to each NP and trace (or, equivalently, gap). This generator generates all possible coindexations between the NP's and their gaps, and among the NP's.[9]

A mechanism such as the "Condition C" mechanism is called a "filter." A filter eliminates the wrong structures that enter it, and pass the correct structures to the next filter or generator. For example, the "Condition C" filter filters out every parse tree that violates the Condition C.

The "Logical Form (LF) Movement" mechanism performs two operations: first, it raises each quantifier (e.g., "every boy", "somebody") and attaches it to the beginning of the innermost sentence (or, equivalently, adjoins it to the nearest IP.) Then, it raises each wh-word (e.g., "who", "where") to the specifier position of the nearest Complementizer Phrase(CP).[10]

To summarize, after the phrase structure is recovered by the LR parser, these structures are passed through a series of filters and generators, until only the correct parses remain. The implementations are taken care of by following a generate and test paradigm (but in a more sophisticated way).

2.2. Subject Binding Generalization

However, this simple first order approximation does not suffice to cover all examples in Korean. Consider these examples:

(5) (i) *ku-ka [**Minswu**-uy emma]-lul coahanta
he-nom Minswu-gen mother-acc like

[7] Also, the mechanism used to avoid "string vacuous scrambling" in Japanese, as described in (Fong, 1994) is used for Korean as well.[8] A "non-vacuous" or "visible" scrambling is a scrambling that "passes over" one or more overt elements (Fong, 1994).

[9] Please refer to (Fong, 1991) for the details on how the free indexation is implemented.

[10] Under the Government and Binding framework, the Complementizer Phrase (CP) immediately dominates the Inflection Phrase (or, equivalently, the Sentence), and wh-words move to the specifier position of a CP at Logical Form (LF) level. The specifier position is immediately dominated by the CP.

'**He** likes **Minswu's** mother.'

(ii) *[**Minswu**-uy emma]$_i$-lul **ku**-ka t_i coahanta[11]

(5)(i) is ruled out by Condition C, since "ku" binds "Minswu", a referring expression. (Please refer to Figure 3 for the definition of "binding."[12]) However, Condition C alone cannot explain why (5)(ii) is unacceptable, since nothing seems to bind "Minswu" (it is therefore free, unbound, and satisfies Condition C.) We can repair this problem by introducing a new definition of binding (defined in (Frank et al., 1992).) According to Lee, (5)(ii) is ruled out by the Subject Binding Generalization:

(6) **Subject Binding Generalization:** If X in subject position binds Y at Deep Structure (D-structure or, equivalently, canonical predicate-argument structure)[13], then X binds Y at all levels of representation (i.e., FOPC level and the surface level).

In (5)(ii), "ku" binds "Minswu" in D-structure, and therefore "ku" still binds "Minswu" in surface structure (S-structure.) Therefore, (5)(ii) is ruled out by Condition C, since "Minswu" is bound.

To implement this, I revised the definitions of binding in the parser. The original and the new definition of binding are shown below as flowcharts.

Here is the original definition of binding (Figure 3):

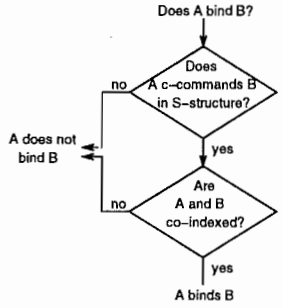

Figure 3: The Original Definition of Binding

Here is the new definition of binding (Figure 4):

[11](Lee, 1994), example 86

[12]A node A c-commands another node B iff the lowest branching node which dominates A also dominates B. For example, in [α [...[β θ]...]], α c-commands β and θ.

[13]Each scrambled element moves back to its canonical position at D-structure.

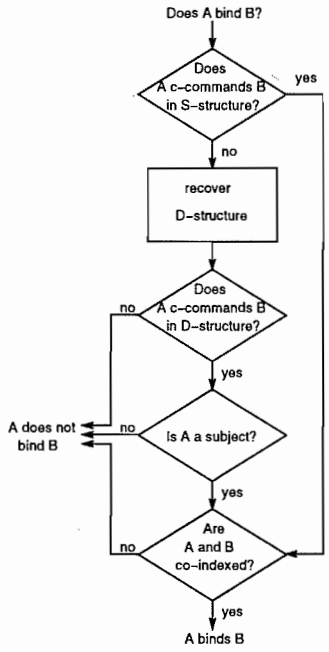

Figure 4: The New Definition of Binding

Notice that a change in the definition of binding automatically changes the definition of Condition C (since the code implementing Condition C calls the code that implements binding.) Also notice that the "subject" is defined as an NP with an "agent" thematic role (θ-role), following Lee's analysis.[14]

Here is how the D-structure is recovered from the S-structure: (1) recurse down the parse tree and replace each element by its D-structure element. For example, a head of a chain (or, equivalently, a filler) would be replaced by an empty element, and a trace (gap) would be replace by its antecedent (filler). (2) Delete any empty elements introduced by step (1).

2.3. Scrambling and Scope

Korean scrambling (as well as scrambling in other languages) is complicated further by examples such as these:

(7) (i) ne-nun [Minswu-ka nwukwu-lul coaha-nunci] a-ni
ne-nun Minswu-nom who-acc like-qm know-qm
'Do you know who Minswu likes?'

(ii) nwukwu$_i$-lul ne-nun minswu-ka t_i coaha-nunci a-ni
'Who do you know Minswu likes?' / 'Do you know who Minswu likes'[15]

In (7), (ii) has a different interpretation than (i). This is because scrambling interacts with scope interpretation. In (Lee, 1994), Lee claims that a scrambled wh-element (like "who" or "what") optionally reconstructs for scope interpretation. Reconstruction means that a scrambled NP optionally moves back to its unscrambled position.

[14] Please refer to (Fong, 1991) for the details on how θ-roles are assigned.

[15] (Lee, 1994), example 160

Remember that at Logical Form (LF) interpretation (or, equivalently, scope interpretation at the FOPC level), the wh-word raises to the specifier position of the *nearest* CP (or Sentence.) The sentence is interpreted as a wh-question if a wh-word occupies the specifier position of the matrix (outer) CP. If a wh-word occupies the specifier position of the embedded (inner) CP, and the specifier position of the matrix (outer) CP is empty, the sentence is interpreted as a yes/no question.

In (7)(i), the wh-word in the embedded (inner) clause "nwukwu" raises to the nearest CP, and the whole sentence is interpreted as a yes/no question, as shown in (8):

(8) (i) [$_{CP}$ [$_{IP}$ ne-nun [$_{CP}$ [$_{IP}$ Minswu-ka *nwukwu* coaha-nunci]] a-ni]]
'The sentence before wh-raising'

(ii) [$_{CP}$ [$_{IP}$ ne-nun [$_{CP}$ *nwukwu*$_i$ [$_{IP}$ Minswu-ka t_i coaha-nunci]] a-ni]]
'*nwukwu* is raised to the specifier position of the nearest CP'

In (8)(ii), wh-word occupies the specifier position of the embedded (inner) CP, and the specifier position of the matrix (outer) CP is empty, i.e., the sentence is interpreted as a yes/no question.

In (7)(ii), the whole sentence can be interpreted as a wh-question (as shown in (10)) as well as a yes/no question (as shown in (9).) For the yes/no interpretation (9), the scrambled wh-word reconstructs to its base position and then raises to the nearest CP.

(9) (i) [$_{CP}$ [$_{IP}$ *nwukwu*$_i$ ne-nun [$_{CP}$ [$_{IP}$ Minswu-ka t_i coaha-nunci]] a-ni]]
'The sentence before wh-raising (*nwukwu* is scrambled to the front of the sentence.)'

(ii) [$_{CP}$ [$_{IP}$ ne-nun [$_{CP}$ [$_{IP}$ Minswu-ka *nwukwu* coaha-nunci]] a-ni]]
'Scrambled *nwukwu* reconstructed.'

(iii) [$_{CP}$ [$_{IP}$ ne-nun [$_{CP}$ *nwukwu*$_i$ [$_{IP}$ Minswu-ka t_i coaha-nunci]] a-ni]]
'*nwukwu* raised to the nearest CP.'

In (9)(iii), a wh-word occupies the specifier position of the embedded (inner) CP, and the specifier position of the matrix (outer) CP is empty, i.e., the sentence is interpreted as a yes-no question.

For the wh-interpretation of (7)(ii), the scrambled wh-word raises to the nearest CP, without undergoing reconstruction.

(10) (i) [$_{CP}$ [$_{IP}$ *nwukwu* ne-nun [$_{CP}$ [$_{IP}$ Minswu-ka coaha-nunci]] a-ni]]
'The sentence before wh-raising (*nwukwu* is scrambled to the front of the sentence.)'

(ii) [$_{CP}$ *nwukwu*$_i$ [$_{IP}$ t_i ne-nun [$_{CP}$ [$_{IP}$ Minswu-ka coaha-nunci]] a-ni]]
'Scrambled *nwukwu* is raised to the nearest CP without undergoing reconstruction.'

In (10)(ii), since a wh-word occupies the specifier position of the matrix (outer) CP, the sentence is interpreted as a wh-question. It is crucial to parse these examples correctly for a Korean Q/A system.

Here is how reconstruction is implemented: (1) Recurse down the parse tree and replace a scrambled wh-word with an empty element, and replace a trace (gap) of a wh-word with its antecedent (filler.) (2) Delete any empty elements introduced by step (1).

Reconstruction is incorporated into the "LF Movement" generator described in Figure 2. The original implementation of the "LF Movement" generator can be understood as two smaller generators serially linked (Figure 5):

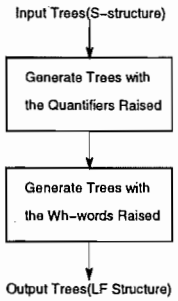

Figure 5: The Original Implementation of the LF Movement Generator

A new definition of the "LF Movement" generator is shown in Figure 6. (This definition will be revised in the following section).

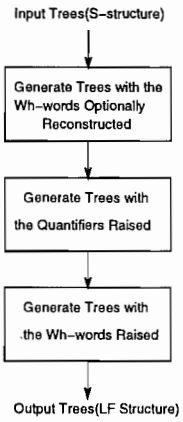

Figure 6: A New Implementation of the LF Movement Generator

Vacuous Wh-Chain Reconstruction The implemented reconstruction algorithm must be more sophisticated if it is to avoid any redundant parses.[16] Here, redundant parses mean that two parses have the same scope interpretations at LF (typed first order predicate calculus) level. Consider this example:

(11) (i) **nwukwu**$_i$-lul [**pro** chinkwu]-ka t_i paypanhayss-ni
who-acc pro-gen friend-nom betrayed-Q
"**Who** did **his** friend betray"[17]

One possibility is to:

1. Reconstruct the chain ($nwukwu_i$,t_i), then
2. Raise "nwukwu" to the nearest CP at LF (or, equivalently, FOPC level.), as shown in (12)

[16]The author would like to acknowledge Dr. Sandiway Fong's help with implementing the algorithms outlined in this subsection (Vacuous Wh-Chain Reconstruction) and the next subsection (Reconstruction for Subject Binding.)
[17](Lee, 1994), example 78b.

(12) (i) $[_{CP}\ [_{IP}$ **nwukwu**$_i$ [**pro** chinkwu]-ka t_i paypanhayss-ni]]
'The sentence before wh-raising (**nwukwu** is scrambled to the front of the sentence.)'

(ii) $[_{CP}\ [_{IP}$ [**pro** chinkwu]-ka **nwukwu** paypanhayss-ni]]
'**nwukwu** is reconstructed.'

(iii) $[_{CP}$ **nwukwu**$_i$ $[_{IP}$ [**pro** chinkwu]-ka t_i paypanhayss-ni]]
'**nwukwu** is raised to the nearest CP after reconstruction.'

Compare this with:

1. No reconstruction, then
2. Raise "nwukwu" from the original scrambled position to the nearest CP at LF, as shown in (13)

(13) (i) $[_{CP}\ [_{IP}$ **nwukwu**$_i$ [**pro** chinkwu]-ka t_i paypanhayss-ni]]
'The sentence before wh-raising (**nwukwu** is scrambled to the front of the sentence.)'

(ii) $[_{CP}$ **nwukwu**$_i$ $[_{IP}$ t_i [**pro** chinkwu]-ka paypanhayss-ni]]
'Scrambled **nwukwu** is raised to the nearest CP without undergoing reconstruction.'

These two parses make no scope distinction, and are therefore redundant. The solution to this "vacuous wh-chain reconstruction" problem is to reconstruct only long distance wh-chains, i.e. only if there is some scope distinction to be derived. This eliminates the first possibility above.
The revised reconstruction algorithm is implemented as follows:

1. Mark each element of a long distance wh-chain, then
2. Replace the trace (gap) with the antecedent (filler), and replace the antecedent (filler) with a null element, for each member of the chain. Mark any null element so introduced for deletion.
3. Delete all elements marked for deletion.

Recall that all reconstruction is optional, so even if the scrambling is long distance, reconstruction may not occur.

Reconstruction for Subject Binding Consider these examples:

(14) (i) [**caki** chinkwu]$_i$-eykey **nwukwuna**-ka t_i komin-ul thelenohnunta
self's friend-dat everyone-nom problem-acc tell.
"**Everyone** tells **his/her** friend problems"[18]

(ii) [**caki** uymwu]$_i$-lul **nwukwuna**-ka t_i chwungsilhi ihaynghayssta
self's duty-acc everyone-nom faithfully carried-out
"**Everyone** carried out **his/her** duty faithfully"[19]

When the quantifier "nwukwuna" is raised at LF, it produces a weak cross-over (WCO) violation.[20] Therefore, we need to avoid WCO violation by reconstructing the scrambled element which is bound by the subject (through the Subject Binding Generalization) before raising the quantifier.

[18] (Lee, 1994), example 79b.
[19] (Lee, 1994), example 80b.
[20] Weak crossover involves the coindexing of an empty category and a genitive inside an NP, as in **Who$_i$ does his$_i$ mother love e_i?* A WCO violation occurs when a wh-word or a quantifier raises from its D-structure position to the [spec, CP] and it "crosses over" a coindexed genitive inside an NP. The presence of a weak crossover makes the sentence unacceptable.

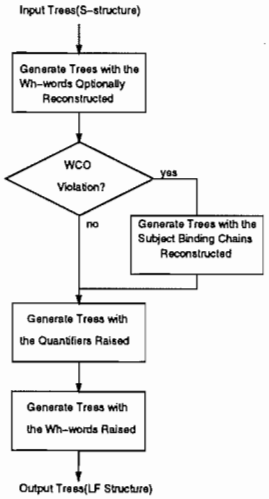

Figure 7: The Final Implementation of the LF Movement Generator

Implementation:

1. Mark each element of a Subject Binding chain, then
2. Replace the trace (gap) with the antecedent (filler), and replace the antecedent (filler) with a null element, for each member of the chain. Mark any null element so introduced for deletion, then
3. Delete all elements marked for deletion.
4. Check that the reconstructed tree satisfies conditions A[21], B[22], and C. If the tree violates any of these conditions, then do not reconstruct (The generator outputs the input tree unchanged.)

The tentative solution is to only allow Subject Binding reconstruction as a last resort for WCO violations.[23]

Implementation Here is the algorithm that implements the ideas outlined above (Figure 7):

[21]Condition A states that an anaphor (e.g., "myself", "herself") must be bound within its governing category. The governing category of an NP is the smallest NP or Inflection Phrase (or, in standard notation, Sentence Phrase) containing that NP, its governor, and an "accessible" subject.

[22]Condition B states that a pronominal (e.g., "he", "they") must not be bound in its governing category.

[23]There is some overlap between the coverage of Subject Binding Generalization and reconstruction of the Subject Binding chain, but they have distinct functions, and both are needed. In the current implementation, Subject Binding Generalization is relevant when analyzing whether the sentence satisfies conditions A, B, and C in the S-structure. Subject Binding reconstruction is applied when the LF-structure is derived from the S-structure. So, without Subject Binding Generalization at the S-structure, reconstruction of the Subject Binding chain may never occur, since the parse tree may have been eliminated before reaching the LF-movement stage. Also, Subject Binding Generalization is relevant for both long and short distance scrambling, but reconstruction of the Subject Binding chain is only (optionally) applicable for long distance chains, as shown above.

3. COMPARISONS WITH OTHER SYSTEMS

To implement the scrambling mechanism described above, less than 150 lines of Prolog code need to be added to the standard **Pappi** framework. Why is scrambling relatively easy to implement in this way? Essentially, **Pappi** can easily handle reconstruction since the code that encodes principles directly deals with sentence structures. In order to handle reconstruction, **Pappi** optionally moves the scrambled wh-elements back to their base position, and then raises them to the nearest CP at logical form interpretation. This is difficult in other systems proposed to handle Korean. First, consider Lin's PRINCIPAR ((Dorr et al., 1995)). Since Lin's PRINCIPAR deals with the *description* of structures, it has difficulty dealing with reconstruction (unless the current design is drastically changed), since the messages it uses only pass *up* parse trees. In order for PRINCIPAR to handle reconstruction, the node representing a trace would have to know whether its antecedent is a wh-word, and decide if it should reconstruct. This would be difficult, since the trace cannot know its antecedent until a message from the trace and the message from the antecedent meet at a node which dominates both the trace and the antecedent. If the scrambled element is going to reconstruct, a message has to travel "down" the tree to the trace, which is not allowed in the current implementation of PRINCIPAR.[24]

Currently, neither PRINCIPAR nor the V-TAG formalism proposed to Korean in (Rambow and Lee, 1994) and (Park, 1994) handle Binding Theory, e.g., Condition C, or Scope Interpretation. Both systems produce the different possible word orders, for both short and long distance scrambling, but neither systems capture the interaction between scrambling and binding, or the interaction between scrambling and scope interpretation.

4. PARSING TIME ANALYSIS

Not surprisingly, scrambling does introduce additional computational complexity into parsing. A sample excerpt from our analysis is given below, where times are normalized to the unscrambled base time. It appears as though multiple scrambling beyond one clause results in the same nonlinear increases observed by Fong (Fong, 1991) for indexing.

	sentence	time, s.	ratio	comment
local	1(a)	1.32	1	(no scrambling)
multiple scrambling	1(b)	2.11	1.60	(one elem scrambled)
	1(c)	1.93	1.46	(one elem scrambled)
	1(d)	3.20	2.42	(two elem scrambled)
	1(e)	2.46	1.86	(one elem scrambled)
	1(f)	3.32	2.52	(two elem scrambled)
long distance	3(a)	6.61	1	(no scrambling)
multiple scrambling	3(b)	8.90	1.35	(one elem scrambled)
	3(c)	15.92	2.41	(two elem scrambled)

5. CONCLUSIONS

This paper describes how one of the most current linguistic analyses of Korean scrambling can be readily incorporated into an existing parser. The parser can correctly handle 26 out of all 29

[24](Lin, p.c.) proposes implementing a filter after the structure has been built, to handle reconstruction, therefore abandoning the structure description idea for this part of the parse.

sets of examples in chapters 2.2 (excluding 2.2.6 on parasitic gaps) and 3.2 of (Lee, 1994). The interaction between scrambling and other components of the grammar is easily accommodated, just as described by Lee. The approach outlined in this paper is compared with other approaches to scrambling, and surpasses them in coverage. The directness with which Lee's linguistic theory can be modeled demonstrates the value of using a "transparent" principles and parameters approach. We can simply use the theoretical assumptions that Lee makes and then test her results using the wide range of scrambling data she exhibits.

ACKNOWLEDGEMENTS

This report describes research done within the Center for Biological and Computational Learning in the Department of Brain and Cognitive Sciences. This research is supported by NSF grant 9217041-ASC and ARPA under the HPCC program.

REFERENCES

Bonnie J. Dorr, Dekang Lin, Jye hoon Lee, and Sungki Suh. 1995. Efficient parsing for korean and english: A parameterized message passing approach. *Computational Linguistics.*

Sandiway Fong. 1991. *Computational Properties of Principle-Based Grammatical Theories.* Ph.D. thesis, MIT, Cambridge, MA 02139.

Sandiway Fong. 1994. Towards a proper linguistic and computational treatment of scrambling: An analysis of japanese. In *Proceedings of COLING-94.*

Robert Frank, Young-Suk Lee, and Owen Rambow. 1992. Scrambling as non-operator movement and the special status of subjects. In *Proceedings of the Third Leiden Conference for Junior Linguists.*

Young-Suk Lee. 1994. *Scrambling as Case-Driven Obligatory Movement.* Ph.D. thesis, University of Pennsylvania, Philadelphia, PA 19104-6228.

Hyun Seok Park. 1994. Korean grammar using tags. Master's thesis, University of Pennsylvania, Philadelphia, PA.

Owen Rambow and Young-Suk Lee. 1994. Word order variation and tree-adjoining grammar. *Computational Intelligence*, 10(4):386–400.

Owen Rambow. 1994. *Formal and Computational Aspects of Natural Language Syntax.* Ph.D. thesis, University of Pennsylvania, 3401 Walnut Street, Suite 400C, Philadelphia, PA 19104-6228.