Implementation of Long-distance Reflexives in Korean

A Categorial Grammar Approach

Yong-hun Lee
Dept. of Linguistics, UIUC
707 S. Mathews Ave. Rm. 4088
Urban, IL 61801, USA
ylee@uiuc.edu

Abstract

This paper provides computational algorithms for a Korean reflexive *caki*, for which both sentence-bound and long-distance readings are possible. Its analyses are based on Chierichia's theory in Categorial Grammar, and a CCG-like system is introduced for the implementation. In this system, we can get both readings of *caki* with the same resolution mechanisms, while the difference is where the reflexive is resolved. These algorithms enable us to account for the distributions and characteristics of a long-distance reflexive *caki* with a more unified way.

1 Introduction

Anaphora has been one of the hottest topics in Linguistics. Various theories have been developed under the name of Binding. There have been roughly two types of Binding Theories. The first group uses tree-theoretic notions to search an antecedent for the anaphora. Chomskyan traditions, especially GB framework, belong to this type of approach. The other utilizes argument-functor relations, instead of structural configurations, for anaphora resolution. This strategy is taken by GPSG (Generalized Phrase Structural Grammar), PLG (Phrase Linking Grammar), and CG (Categorial Grammar). Chierchia (1988)'s theory in Categorial Grammar is a good example.

There have been lots of observations that the distributions of reflexives are different from those of pronominals. That is, reflexives have some more restrictions that pronominals do not obey. In order to implement this intuition, many theories adopted two kinds of different mechanisms, one for reflexives and the other for pronominals. The conditions by which reflexives are licensed are different from those for pronominals.

Korean reflexives have characteristics that English counterparts do not exhibit. In Korean, long-distance reflexives are possible in addition to ordinary sentence-bound ones. The distributions of *caki* in (1), (2), and (3) illuminate this fact[1,2].

(1) ***Chelsoo***$_1$*-ka* ***caki***$_1$*-lul* *salangha-n-ta.*
Chelsoo.NOM self.ACC love.PRES.DECL
'Chelsoo loves himself.'

---

[1] Korean has another kind of reflexive that permits only sentence-bound reading. *Cakicasin* in (i) is an example.

(i) ***Chelsoo***$_1$*-nun* [***Younghee***$_2$*-ka* ***cakicasin***$_{*1/2}$*-ul* *salangha-n-ta-ko*] *sayngkakha-n-ta.*
Chelsoo.TOP Younghee.NOMself.ACC love.PRES.DECL.COMP think.PRES.DECL
a. *Chelsoo thinks that Younghee loves him.
b. Chelsoo thinks that Younghee loves herself.

In (i), *cakicasin* can refer to only *Younghee*, not *Chelsoo*. This paper does not deal with this sentence-bound reflexive *cakicasin*. Instead, it focuses on two readings of a long-distance reflexive *caki*. But we can provide resolution algorithms for *cakicasin* by slight modifying those for *caki*. For the different distributions and resolution algorithm for *cakicasin*, see Lee (2001b).

[2] According to Moon (1996) and others, long-distance reading (3a) is superior to sentence-bound interpretation in (3b) for the sentence.

(2) ***Chelsoo***$_1$*-nun* [***caki***$_1$*-ka* *ttokttokha-ta-ko*] *sayngkakha-n-ta.*
Chelsoo.TOP self.NOM be smart.DECL.COMP think.PRES.DECL
'Chelsoo thinks that he is smart.'

(3) ***Chelsoo***$_1$*-nun* [***Younghee***$_2$*-ka* ***caki***$_{1/2}$*-lul* *salangha-n-ta-ko*]
Chelsoo.TOP Younghee.NOM self.ACC love.PRES.DECL.COMP
sayngkakha-n-ta.
think.PRES.DECL
a. Chelsoo thinks that Younghee loves him.
b. Chelsoo thinks that Younghee loves herself.

In (1), *caki* is sentence-bound, i.e., the antecedent of *caki* is within the sentence boundary. In (2), *Chelsoo*, an antecedent, is outside of the sentence boundary; and it shows a case of long-distance reflexive. In (3), both sentence-bound reading and long-distance interpretation are possible for *caki*. Sentences in (4) and (5) correspond to English counterparts of (2) and (3), where long-distance interpretations are impossible.

(4) a. *Chelsoo thinks that himself is smart.
b. Chelsoo thinks that he is smart.

(5) a. *Chelsoo thinks that Younghee loves himself.
b. Chelsoo thinks that Younghee loves herself.

The goal of this paper is to provide computational algorithms for the interpretations of Korean reflexives, for which both sentence-bound and long-distance readings are possible. For the analyses, a CCG-like system will be introduced, which is slightly modified from Steedman (1996, 2000)'s Combinatory Categorial Grammar (CCG). Through the analyses, this paper will show how two different interpretations of Korean reflexives are calculated and how they are implemented.

2 Binding Theory in Categorial Grammar

Categorial Grammar was first introduced by Ajdukiewicz (1935) and later modified and advanced by Bar-Hillel, Curry, and Lambek. In this framework, we have two basic categories *n* and *s*, and other categories come from the combinations of these two categories. All the syntactic phenomena are described and analyzed by the functor-argument relations of categories.

Steedman (1996, 2000) extended previous studies in Categorial Grammar and developed Combinatorial Categorial Grammar (CCG). The most important characteristic of his system is that predicates-argument relations are projected by the combinatory rules of syntax, and other operations are based on these relations (Steedman, 2000:38). Two basic combinatory rules are *functional application* and *functional composition*, which are delineated in (1).

(6) Two Basic Combinatory Rules
a. Functional Application
(i) X / Y Y ⇒ X
(ii) Y X / Y ⇒ X
b. Functional Composition
(i) X / Y Y / Z ⇒ X / Z
(ii) X / Y Y / Z ⇒ X / Z
(iii) Y / Z X / Y ⇒ X / Z
(iv) Y / Z X / Z ⇒ X / Z

Chierchia took Categorial Grammar to explain Binding phenomena in English, and he described syntactic constraints of reflexives and pronominals as follows.

(7) Binding Theory in Categorial Grammar (Chierichia, 1988:134)
a. A reflexive must be bound to an F-commanding argument in its minimal NP or S domain.
b. A non-reflexive pronoun must not be co-indexed with anything in its minimal NP or S domain.
where F-command is simply c-command at function-argument structure.[3]

Agreement in *number* and *gender* must hold between pronouns and their antecedents, in order to pronouns can refer to their antecedents. The constraint for checking agreement is stated in (8a). FT(*n*) in (8b) has three information: *n* is the index of the given NP, *gndr* is gender, and *nmbr* is number.

(8) Agreement between Antecedent and Pronouns (Chierchia, 1988:132)
a. FT(*n*)≈FT(*m*): The features associated with *n* are non-distinct from those associated with *m*.
b. $FT(n) = \begin{bmatrix} n \\ \text{gndr} \\ \text{nmbr} \end{bmatrix}$

For example, the FTs of three different NPs *John*, *himself*, and *her* can be stated as follows[4].

(9) $FT(1) = \begin{bmatrix} \text{John}_1 \\ 1 \\ \text{male} \\ 3 \end{bmatrix}$ $FT(2) = \begin{bmatrix} \text{himself}_2 \\ 2 \\ \text{male} \\ 3 \end{bmatrix}$ $FT(3) = \begin{bmatrix} \text{her}_3 \\ 3 \\ \text{female} \\ 3 \end{bmatrix}$

Chierchia developed resolution algorithms for pronouns based on the combinatory rules of categories, and it can be stated as in (10).

(10) Chierchia's Algorithms (1988:138-9)
a. $\underset{0}{TV} + \underset{1}{NP} \Rightarrow \underset{2}{IV}$ (here and throughout integers will be used as names for the categories mentioned in the rules)

conditions: (i)	$LPS(0) \cap LPS(1) = \varnothing$[5]	non-coreference
(ii)	$SLASH(2) \cap (LPS(1) \cup LPS(2)) = \varnothing$	crossover
(iii)	$SLASH(2) = SLASH(0) \cup SLASH(1)$	slash-percolation
(iv)	$LPS(2) = LPS(0) \cup LPS(1)$	LPS-percolation

b. $\underset{0}{S/NP_n} + \underset{1}{NP_n} \Rightarrow \underset{2}{S}$

conditions: (i)	$LPS(2) = \varnothing$	A-opacity boundary
(ii)	$SLASH(2) \cap (LPS(1) \cup LPS(2)) = \varnothing$	crossover
(iii)	$SLASH(2) = SLASH(0) \cup SLASH(1)$	slash-percolation
(iv)	$n \notin \underset{+refl}{} LPS(0) \cup LPS(1)$	reflexive

---

[3] This condition has different prediction for the sentences in (i) and (ii). It rules in (i), but rules out (ii) (Chierchia, 1988:135).

(i) Mary showed the men each other.
(ii) *Mary showed each other the men.

Binding Theory in GB says that both sentences are grammatical because *each other* and *the men* c-command each other, which is wrong.

[4] Note that *John*, *himself*, and *her* stand for an R-expression, a reflexive, and a pronominal respectively.

[5] LPS (Local Pronoun Store) stores indices for pronouns and their antecedents. See Section 4.1.

c. $\underset{0}{IV}/\underset{1}{IV} + \underset{1}{IV} \Rightarrow \underset{2}{IV}$

conditions: (i) LPS(2) = LPS(0) — A-opacity boundary
(ii) SLASH(2) ∩ (LPS(1) ∪ LPS(2)) = ∅ — crossover
(iii) SLASH(2) = SLASH(0) ∪ SLASH(1) — slash-percolation
(iv) $n \notin_{+refl}$ LPS(1) — reflexive

d. Reflexives

(i) $\underset{n\,[+refl] \in LPS}{A} \Rightarrow \underset{n \notin LPS}{A}$

(ii) conditions: (a) A = IV, TV, (b) FT(A) ≈ FT(n)

(iii) translation: $\lambda x_n\,[A'(x_n)]$

According to these algorithms, two pronouns *himself* and *him* are resolved as in (11) and (12). Here note that the translations of two sentences are *love'*$(x_2)(x_2)$ and *love'*$(x_1)(x_2)$.

(11) He loves himself.

<he loves himself, S, love'$(x_2)(x_2)$, LPS:∅> (ignoring SLASH)

<he, NP_2, x_2, LPS:2> <love himself, S/NP_2, λx_3 [love'$(x_3)(x_3)$], LPS:2>

<love himself, S/NP_2, love'(x_3), LPS:2, 3_{+ref}>

<love, $(S/NP_2)/NP^6$, love', LPS:2><himself, NP_3, x_3, LPS:3_{+ref}>

(12) He loves him.

<he loves him, S, love'$(x_1)(x_2)$, LPS:∅> (ignoring SLASH)

<he, NP_2, x_2, LPS:2> <love him, S/NP_2, love'(x_1), LPS:1, 2>

<love, $(S/NP_2)/NP$, love', LPS:2><he, NP_1, x_1, LPS:1>

Because *himself* is a reflexive, (10d) is applied in the analysis in (11), and changes *love'*(x_3) into λx_3 [*love'*$(x_3)(x_3)$]. This operation does not occur in (12) because *him* is a pronominal, which results in different translations between reflexives and pronominals.

3 Resolving Long-distance Reflexives in Korean

Chierichia's algorithms are designed to handle English data. But, in order to explain the distributions of long-distance reflexives in Korean, some conditions must be modified. First, According to the condition (10bi), LPS value of every S node must be ∅. To get the long-distance readings in (2) and (3), however, this condition must not be applied, because it prevents (2) and (3) have only sentence-bound interpretations. Then, (2) is ruled out and *caki* in (3) refers to only *Younghee*, which are counterintuitive. Second, as the sentence in (3) illustrates, *caki* can refer to not only male entity but also female entity. As Hong (1985) and Kang (1988b) have already pointed out, *gender* information is not crucial for reflexive resolution in Korean. Therefore we have to make the algorithms insensitive to *gender* information of the specification in (8b). The agreement information that (8b) contains are *index* of the NPs, *person*, and *number*. After the modifications of these two conditions, we can analyze (1) and (3) as in (13) and (14) respectively. Here, note that the translation of a complementizer *-ko* is specified as $\lambda\phi[\hat{}\,\phi]$.

[6] Here 2 in $(S/NP_2)/NP$ means the index of subject NP must be 2. It is a tool for subject-predicate agreement.

(13)a. *Chelsoo*$_1$*-ka* *caki*$_1$*-lul* *salangha-n-ta.*
Chelsoo.NOM self.ACC love.PRES.DECL
'Chelsoo loves himself.'

b. <Chelsoo-ka caki-lul salangha-n-ta, S, love'(x_2)(x_2), LPS:∅>

<Chelsoo-ka,NP_2,x_2,LPS:2><caki-lul salangha-n-ta,S/NP_2,λx_3[love'(x_3)(x_3)],LPS:2>

<caki-lul salangha-n-ta, S/NP_2, love'(x_3), LPS:2, 3_{+ref}>

<caki-lul, NP_3, x_3, LPS:3_{+ref}><salangha-n-ta, (S/NP_2)/NP, love', LPS:2>

(14)a. [1]*Chelsoo*$_1$*-ka* [[2]*Younghee*$_2$*-ka* [3]*caki*$_{1/2}$*-lul* [4]*salangha-n-ta-*[5]*ko*]
Chelsoo.NOM [Younghee.NOM self.ACC love.PRES.DECL].COMP
[6]*sayngkakha-n-ta.*
think.PRES.DECL

b. (i) 'Chelsoo thinks that Younghee loves him.'
(ii) 'Chelsoo thinks that Younghee loves herself.'

c. <[1]+[2]+[3]+[4]+[5]+[6], S, think'(c, ^love'(y,c), LPS:∅>

<[1],NP_1,c,LPS:1><[2]+[3]+[4]+[5]+[6],S/NP_1,λx_3[think'(x_3,^love'(y,x_3))],LPS:1>

<[2]+[3]+[4]+[5]+[6], S/NP_1, think'(^love'(y,x_3)), LPS:1, 3_{+ref}>

<[2]+[3]+[4]+[5],S',^love'(y,x_3),LPS:3_{+ref}><[6],(S/NP_1)/S',think',LPS:1>

<[2]+[3]+[4],S,love'(y,x_3),LPS:3_{+ref}> <[5],S'/S,$\lambda\phi$[^ϕ],LPS:∅>

<[2], NP_2, y, LPS:2> <[3]+[4], S/NP_2, love'(x_3), LPS:2, 3_{+ref}>

<[3], NP_3, x_3, LPS:3_{+ref}> <[4], (S/NP_2)/NP, love', LPS:2>

d. <[1]+[2]+[3]+[4]+[5]+[6], S, think'(c, ^love'(y,y)), LPS:∅>

<[1], NP_1, c, LPS:1><[2]+[3]+[4]+[5]+[6], S/NP_1, think'(^love'(y,y)), LPS:1>

<[2]+[3]+[4]+[5],S',^love'(y,y),LPS:∅><[6],(S/NP_1)/S',think',LPS:1>

<[2]+[3]+[4], S, love'(y,y), LPS:∅> <[5], S'/S, $\lambda\phi$[^ϕ], LPS:∅>

<[2], NP_2, y, LPS:2> <[3]+[4], S/NP_2, λx_3[love'(x_3,x_3)], LPS:2>

<[3]+[4], S/NP_2, love'(x_3), LPS:2, 3_{+ref}>

<[3], NP_3, x_3, LPS:3_{+ref}> <[4], (S/NP_2)/NP, love', LPS:2>

In (13), the reflexive *caki* is resolved within the S node, where it finds the antecedent *Chelsoo*. (14) has two S nodes, and both *Chelsoo* and *Younghee* can be an antecedent for *caki*. In (14c), the reflexive rule (10d) is applied in the higher S/NP node, whereas it is applied to lower S/NP node in (14d). Likewise, the reflexive *caki* is resolved in the higher S nodes in (14c), while it finds an antecedent in the lower S node in (14d). As the translations illustrate, (14c) corresponds to long-distance reading in (14bi) and (14d) to sentence-bound reading of (14bii) respectively. With some modifications of Chierchia's idea, we can correctly get the both interpretations for *caki* as above.

4 Computational Implementation

4.1 A CCG-like System

There are three possible ways of implementing the analyses in (13) and (14). One is to use translation, which is represented by predicate logic. Another method is to utilize LPS and SLASH values, where the indexes of NP are stored. Lee (2001a) employed this technique. The other option is to combine the operations on translations and LPS & SLASH values. This paper chooses the last one. That is, reflexives are resolved by the combined operations of translation and LPS value of each node. But SLASH will be ignored as above, because this paper does not handle crossover phenomena.

The system that this paper develops is a CCG-like system[7]. It is the combination of Chierchia's ideas into Steedman's Combinatory Categorial Grammar. This system is similar to Steedman's system in that surface combinatorics triggers other operations, especially reflexive resolution algorithms. It is different from Steedman's in the following two points. First, PS rules are also important in this system, while Steedman did not utilize it. Second, it has five attributes to describe syntactic dependencies, which resembles those in HPSG.

In the implementation, we will have five attributes, PHON, CAT, AGR, TRANS and LPS. PHON concatenated a word to a stream of words. CAT has categorial information, such as S, N, S/N, and so on. AGR is for agreement information, *person* and *number*. TRANS has the translation of the node. LPS (Local Pronoun Store) is something like a Cooper-storage, but it stores indices for each NP. The functional application on the CAT value triggers operations on TRANS and LPS values, and all the reflexives, whether they have sentence-bound or long-distance reading, are resolved by these operations.

4.2 Syntactic Process

In this paper, we will use a Korean fragment, and the PS rules for them are enumerated in (15).

(15) PS Rules for a Korean Fragment
1. S → NP VP
2. NP → N
3. NP → PRON
4. VP → V
5. VP → NP V
6. VP → S' V
7. S' → S C

Syntactic processes have two important roles in the system. First, they decide which constituent and which one, among the stream of strings, go into the predicate-argument relations. Second, they filter out ungrammatical sentences when a Case mismatch occurs. For example, they exclude the sentences like followings.

(16) a. *Her loves himself.
b. *He loves she.

While Park (2001a, 2001b) and Lee & Park (2001) rule out them by category combinations themselves, these sentences cannot be an input to the following operations because they are filtered out in the syntactic process in this system.

---

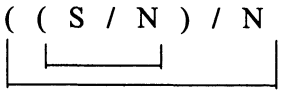
[7] A different, but related, version of CCG-like system is developed by Park (2001a, 2001b) and Lee & Park (2001). Their focus is how Korean Case markers can be handled in Categorial Grammar. Because this paper concernes only long-distance reflexives, it will not illustrate how Case markers can be dealt with in Categorial Grammar. For a theoretical approach and its importance, see Park (2001a, 2001b). For its computational implementation, see Lee & Park (2001).

4.3 Operations on Category[8]

Before we start to develop algorithms for reflexive resolution in Korean, we need two more mechanisms for handling categories. One is recognizing and separating categories, and the other is to provide implementation for functional application.

A category has a form A or A / B. When we have an atomic category N or S, what we have to do is just returning the categories. When we meet a complex category such as ((S/N)/N)/N, we need to separate this one into two categories, (S/N)/N and N. The important clues for separating categories are the nested structure of parentheses and a slash ('/') between two categories. That is, our example category can be analyzed as in (17).

(17) Nested Structure of Parentheses



In its computational implementation, a stack meets our purpose. A stack is a LIFO-style storage, where LIFO means Last In First Out. We have two operations PUSH and POP for the stack. PUSH stores something into the stack, whereas POP removes the top-most element from the stack. In its computational implementation, we PUSH a left parenthesis ('(') into the stack when we meets it, and POP a left parenthesis when we meets a right parenthesis (')'). But we don't need to use a real stack, rather we can use a virtual stack. A virtual stack can be implemented just by increasing or decreasing the pointer of the stack, because all the contents of the stack are identical. The concrete algorithm is illustrated in (18).

(18) Algorithm for Separating a Category

```
function string separate_category (string s)
var
   string A, B;
begin
   if there is no / in s then return s;
   else if there is one / and no ( in s then
      begin
         put the string before / into A;
         put the string after / into B;
         return(A, B);
      end
   else
      begin
         while read through s do
            begin
               if there is a ( then PUSH('(');
               else if there is a ) then POP('(');
            end;
         if the stack is empty then put the string up to now into A;
         if the next string is not / then return error;
         while read through s do
            begin
               if there is a ( then PUSH('(');
               else if there is a ) then POP('(');
            end;
         if the stack is empty then put the string up to now into B;
         if the stack is empty then return(A, B) else return error;
      end;
end;
```

[8] This part comes from Lee (2001a) and Lee & Park (2001). Because the operations themselves are the same, whether it is Korean or English, this paper uses the same algorithms.

The algorithm for functional application is as in (19). In func_application(s_1, s_2), s_1 is a functor and s_2 is an argument. This function checks if s_2 is the argument of s_1. If it is, the function returns the result of functional application. If not, it returns an error message.

(19) Algorithm for Functional Application

```
function string func_application (string s1, string s2)
var
   string A, B;
begin
   A, B = separate_category(s1);
   if s2 = B then return A else return error;
end;
```

When s_1 is an argument of s_2, the position of s_1 and s_2 is interchanged outside of this function and functional application is tried again.

4.4 Reflexive Resolution Algorithms

To develop algorithms for long-distance reflexives, let's scrutinize the analyses in (14) again.

(14) a. [1]*Chelsoo*$_1$*-ka* [2]*Younghee*$_2$*-ka* [3]*caki*$_{1/2}$*-lul* [4]*salangha-n-ta-*[5]*ko*]
Chelsoo.NOM [Younghee.NOM self.ACC love.PRES.DECL].COMP
[6]*sayngkakha-n-ta.*
think.PRES.DECL

b. (i) 'Chelsoo thinks that Younghee loves him.'
(ii) 'Chelsoo thinks that Younghee loves herself.'

c. <[1]+[2]+[3]+[4]+[5]+[6], S, think'(c, ^love'(y,c), LPS:∅>

<[1],NP$_1$,c,LPS:1><[2]+[3]+[4]+[5]+[6],S/NP$_1$,λx_3[think'(x_3,^love'(y,x_3))],LPS:1>

<[2]+[3]+[4]+[5]+[6], S/NP$_1$, think'(^love'(y,x_3)), LPS:1, 3$_{+ref}$>

<[2]+[3]+[4]+[5],S',^love'(y,x_3),LPS:3$_{+ref}$><[6],(S/NP$_1$)/S',think',LPS:1>

<[2]+[3]+[4],S,love'(y,x_3),LPS:3$_{+ref}$> <[5],S'/S,$\lambda\phi$[^ϕ],LPS:∅>

<[2], NP$_2$, y, LPS:2> <[3]+[4], S/NP$_2$, love'(x_3), LPS:2, 3$_{+ref}$>

<[3], NP$_3$, x_3, LPS:3$_{+ref}$> <[4], (S/NP$_2$)/NP, love', LPS:2>

d. <[1]+[2]+[3]+[4]+[5]+[6], S, think'(c, ^love'(y,y)), LPS:∅>

<[1], NP$_1$, c, LPS:1><[2]+[3]+[4]+[5]+[6], S/NP$_1$, think'(^love'(y,y)), LPS:1>

<[2]+[3]+[4]+[5],S',^love'(y,y),LPS:∅><[6],(S/NP$_1$)/S',think',LPS:1>

<[2]+[3]+[4], S, love'(y,y), LPS:∅> <[5], S'/S, $\lambda\phi$[^ϕ], LPS:∅>

<[2], NP$_2$, y, LPS:2> <[3]+[4], S/NP$_2$, λx_3[love'(x_3,x_3)], LPS:2>

<[3]+[4], S/NP$_2$, love'(x_3), LPS:2, 3$_{+ref}$>

<[3], NP$_3$, x_3, LPS:3$_{+ref}$> <[4], (S/NP$_2$)/NP, love', LPS:2>

As we can find from the analyses, the resolution of reflexives takes two steps. The first step is the reflexive rules in (10d) is applied to the S/NP nodes, whose operations are (i) to delete the index for the reflexive NP from LPS and (ii) to change the translation into a λ-expression. The second step is, which is occurs in the S nodes, (i) to find a possible antecedent for the reflexive, (ii) to check the agreement between the reflexive and the antecedent, (iii) to resolve the reflexive, and (iv) to update the result by λ-conversion of the translation. But we need one more operation in addition to those two. We need to decide when we resolve the reflexive. That is, *caki* in (14c) is resolved with *Chelsoo* while it is paired with *Younghee* in (14d). An algorithm is necessary for deciding the point where the reflexive is resolved. These three mechanisms are the crux of resolution algorithms.

The first step can be easily implemented, because its mechanisms are straightforward. The algorithm for them is as follows.

(20) Algorithms for Reflexive Rule

```
function avop reflexive_rule(avop i, int j)[9]
var
    string iTRANS, iVAR, rTRANS;
    set iLPS, rLPS;
begin
    iLPS := LPS(i);
    iLPS := iLPS – j;
    iTRANS := TRANS(i);
    iVAR := x_j;
    rTRANS := λ + iVAR + '[' + iTRANS + '(' + iVAR + ')' + ']';
    update i with rTRANS and rLPS;
end;
```

Here, j is the index of the reflexive NP_j. The first two lines find the index for reflexive and delete it from LPS. The next three lines take the TRANS value, find the variable x_j, and turn the TRANS into a λ-expression. Then the algorithm updates the attribute-value ordered pair with the new values.

The second step is to find a possible antecedent, and resolve the reflexives if agreement information between the pronoun and the antecedent matches. The algorithm for this mechanism can be described as follows.

(21) Algorithms for Reflexive Resolution

```
function avop reflexive_resolution(avop i, avop j, avop k)
var
    string fPHON, sPHON, iTRANS, rTRANS;
begin
    fPHON := PHON(j);
    sPHON := PHON(k);
    iTRANS := TRANS(i);
    rTRANS := lamda_conversion(iTRANS);
    print(fPHON + " refers to " + sPHON + ".");
    update i with rTRANS;
end;
```

Here, i is the ordered pair of S node, j is the index of the reflexive NP_j, and k is the index of the antecedent NP_k whose *person* and *number* matches with those of NP_j. iTRANS is the translation of S node, and it has a form λ x_j [... (x_j) ...] (TRANS(k)), where λx_j[...(x_j)...] comes from VP, and TRANS(k) from subject NP_k. Lamda conversion changes iTRANS into [... (TRANS(k)) ...]. The agreement between NP_j and NP_k is checked outside of reflexive_resolution(), and this function is called when two NPs match in their *person* and *number*, where NP_k is the antecedent of NP_j.

[9] Here, **avop** means an attribute-value ordered pair, which is <PHON,CAT,AGR,TRANS,LPS>.

The last algorithm for reflexive resolution is the one who decides when the reflexive is resolved. The algorithm can be stated as follows. Before the system begins to analyze the sentence, the attribute-value ordered pairs of each lexical item are loaded into the main memory. For example, the lexical items for (14a) are stored in the main memory as follows.

(22) Ordered Pairs of Each Lexical Items in (14a)[10]

01: <Chelsoo-ka,n,M3S,c,LPS:{0[+name]}>
02: <Younghee-ka,n,F3S,y,LPS:{1[+name]}>
03: <caki-lul,n,3S,x2,LPS:{2[+refl]}>
04: <salangha-n-ta,(s/n)/n,N/A,love',LPS:{1}>
05: <ko,s'/s,N/A,$\lambda\phi[\hat{}\phi]$,LPS:{}>
06: <sayngkakha-n-ta,(s/n)/n,N/A,think',LPS:{0}>

These ordered pairs are used during the syntactic process and the analyses. Before the algorithms start to analyze the sentences, a variable *n* is set to point the first position, i.e. 01, in these ordered pairs.

The algorithm goes as follows. When the analysis starts, the current position of *n* is 01. That is, *n* is pointing *Chelsoo-ka* right now. When we meets a reflexive NP_j *caki*, the algorithm checks if the *person* and *number* information of two NPs, NP_j and NP_n, agree. If they agree the index of *caki* is percolated up until it meets *Chelsoo-ka*. When the reflexive meets with a S node whose subject is NP_n *Chelsoo-ka*, the algorithms call the function reflexive_rule() in the VP node, then call reflexive_resolution() in the S node, where the reflexive is resolved. If the agreement value of *n* does not match with that of the reflexives, *n* is increased by 1 and the agreement attribute is checked again. After the reflexive is resolved, the algorithm prints out the result, i.e. '*caki refers to Chelsoo*', and deletes all the result except those in (22). Then, *n* is increased, and the same procedures are performed again. All the procedures end when *n* becomes equal to the index for the reflexive, for example, $n = 3$ for (22).

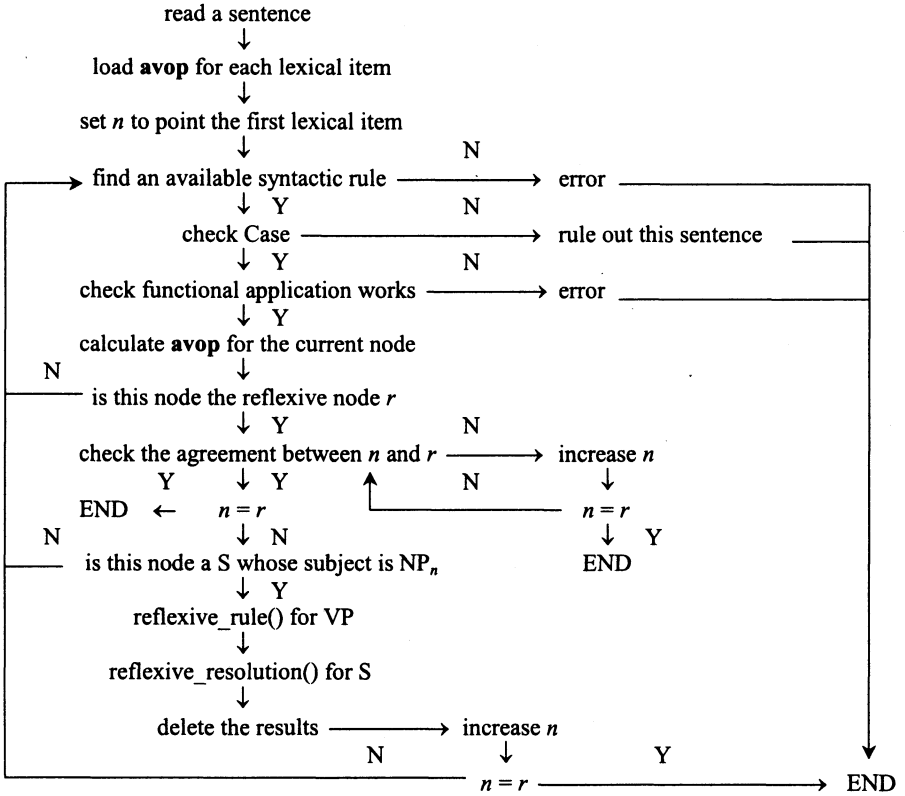
By those mechanisms, we can correctly get the results, '*caki refers to Chelsoo*' and '*caki refers to Younghee*'.

4.5 Overall Flowchart

Now that we have all the algorithms for each subpart, it's time to view the whole picture. The overall flowchart is as in (23).

[10] In fact, two more attributes, POS and CASE, must be added to these ordered pairs. POS (Part Of Speech) is for the syntactic process that is described in Section 4.2, and CASE is used to rule out the sentences in (16). Because this paper is not focused on such syntactic processes, those attributes are omitted in the representations.

(23) Overall Flowchart for Reflexive Resolution in Korean



When the reflexive is resolved, the algorithm delete all the results and goes to the point where it finds the next available syntactic rule. Because everything else except attribute-value ordered pairs for lexical items is deleted, the procedure starts from the nearly first stages, just except that *n* is updated. With this technique, we can get the right results for Korean reflexives.

5 Advantages of a CCG-like System

As mentioned in Section 1, there are roughly two kinds of Binding Theories. One uses structural configurations and the other utilizes functor-argument relations. As Kang (1988a, 1988b, 2001) have already pointed out, the second approaches are better than the first ones in explaining distributions and characteristics long-distance reflexives. To enumerate the advantages of the latter methods deserves another paper. As an example, Kang (1988b) investigated long-distance reflexives in Korean, Japanese, Swedish, and Icelandic, and illustrated that the latter strategies can efficiently account for the examples that are problematic for configuration-based explanations. Among the advantages that he pointed out, the most important one for this CCG-like system is that we don't need to have two separate domains for reflexive resolution. As the analyses in (13) and (14) demonstrate, whether the reflexives are resolved within the minimal S node or outside of the S node, the resolution mechanisms themselves are the same, i.e., by the reflexive rule and lamda conversion. If we try to implement reflexive resolution by configuration, such as in Hong & Lee (1994), we must set different Binding Domains under which the reflexives find their antecedents, one for sentence-bound reading the other for long-distance interpretation.

From the implementational point of view, the advantage of a CCG-like system is that we don't need a parse tree for the resolution of reflexives. The systems that depend on structural configurations presuppose the existence of a parse tree, such as that of Lee (1996a, 1996b). In this system, however, PS rules and functional applications take the roles of a parser, and all the operations depend on argument-functor relations. The PS rules in (15) and syntactic processes in Section 4.2 DO NOT make any parse tree. Their functions are (i) to decide which constituent and which one go into functor-argument relations, and (ii) to filter out ungrammatical sentences in (16). Therefore, we have no parse tree in this CCG-like system. In a nutshell, we achieve the same effect of a parser without it, and

this characteristic speeds up the system because we don't need to spend time for constructing a parse tree.

6 Conclusion

This paper developed resolution algorithms for Korean reflexives within Categorial Grammar framework. Those algorithms are based on the functor-argument relations of the constituents, rather than depend on structural configurations. A CCG-like system is introduced where surface combinatorics plays an important role. Each reflexive is resolved by the operations on the TRANS and LPS values, where each operation is triggered by the functional application of CAT values. We found that long-distance reflexives in Korean are efficiently accounted for with those mechanisms. We also saw that resolution algorithms using functor-argument relations are better than configuration-based accounts. Theoretically, we can explain two readings of long-distance reflexives by the same resolution algorithms, while the difference of two interpretations is captured with different resolution points. In implementation, we said that the algorithms speed up the processing time, because we don't need a parser.

In sum, this paper provides a faster and succinct computational implementation for the interpretations of long-distance reflexives in Korean, within Categorial Grammar. I hope this study can give us an opportunity to understand characteristics of long-distance reflexives in Korean.

References

Chierchia, G. 1988. Aspects of a Categorial Theory of Binding. In R. Oehrle et al. eds., *Categorial Grammars and Natural Language Structures*. 125-51. Dordrecht : R. Reidel.

Hong, Sungshim. 1985. *A and A' Binding in Korean and English : Government-Binding Parameters*. Ph.D. Dissertation. University of Connecticut.

Hong, Sungshim and Yong-hun Lee. 1994. On the Implementation of Pronominal Readings Using Binding Condition B. *Proceedings of the Korean Society for Cognitive Science*. 443-459.

Kang, Beom-Mo. 1988a. *Functional Inheritance, Anaphora, and Semantic Interpretation in a Generalized Categorial Grammar*. Ph.D. Dissertation. Brown University.

_____. 1988b. Unbounded Reflexives. *Linguistics and Philosophy* 11:415-456.

_____. 2001. *Categorial Grammar : Korean Morphology, Syntax, and Type-theoretic Semantics*. Written in Korea. Seoul : Korea University.

Lee, Yong-hun. 1996a. On Efficient Parsing Method Using X-bar Theory. *Proceedings of the Korean Society of Cognitive Science*. 33-51.

_____. 1996b. *On the Implementation of English Anaphor and Pronominal Interpretation Using Binding Conditions and Conceptual Structures*. MA Thesis. Taejon : Chungnam National University.

_____. 2001a. Implementation of Pronoun Readings in English : A Categorial Grammar Approach. A Paper Presented at the Seoul International Conference on English Linguistics. Yonsei University, Seoul, Korea.

_____. 2001b. Sentence-bound vs. Long-distance Reflexives in Korean : A Categorial Grammar Approach. Manuscript. UIUC.

Lee, Yong-hun and Chongwon Park. 2001. An Analysis of Korean Multiple Accusative Constructions and Its Computational Implementation. A Paper Presented at the Annual Conference of Linguistic Society of Korea, Kyung-Joo, Korea.

Moon, Seung-Chul. 1996. *An Optimality Approach to Long Distance Anaphors*. Seoul : Thaehaksa.

Park, Chongwon. 2001a. Semantic and Combinational Valences for Korean Case Markers and COMPs: A Bi-directional Analysis and Two Types of Functor-Argument Relations. to appear in *Japanese-Korean Linguistics XI*.

_____. 2001b. Two Types of Valence and Tow Types of Functor-Argument Relations. to appear in *Harvard Studies in Korean Linguistics IX*.

Steedman, M. 1996. *Surface Structure and Interpretation*. Cambridge, MA : MIT Press.

_____. 2000. *The Syntactic Process*. Cambridge, MA : MIT Press.