The Institute For Research In Cognitive Science

Korean Grammar Using TAGs

by

Hyun Seok Park

University of Pennsylvania
3401 Walnut Street, Suite 400C
Philadelphia, PA 19104-6228

December 1994

Site of the NSF Science and Technology Center for
Research in Cognitive Science

University of Pennsylvania
Founded by Benjamin Franklin in 1740

IRCS Report 94-28

UNIVERSITY OF PENNSYLVANIA

THE MOORE SCHOOL OF COMPUTER AND INFORMATION SCIENCE

SCHOOL OF ENGINEERING AND APPLIED SCIENCE

KOREAN GRAMMAR USING TAGS

HYUN SEOK PARK

Philadelphia, Pennsylvania

December, 1994

A thesis

presented to the Faculty of Engineering and Applied Science of the University of Pennsylvania in partial fulfillment of the requirements for the Degree of Master of Science in Engineering for graduate work in Computer and Information Science.

Aravind Joshi

(Advisor)

Mark Steedman

(Graduate Group Chairperson)

0

© Copyright 1995

by

Hyun Seok Park

Acknowledgements

First, I thank my advisor, Dr. Aravind K. Joshi and Group Chairman Dr. Mark Steedman for their constant interest and encouragement.

I express my deepest appreciation to Dr. Martha Palmer and Dania Egedi for their enormous help and friendship in every aspect.

My parents and sister receive my most special thanks, who encouraged me, and stood steadily by me. It is hard to understand why people sacrifice themselves for somebody else. It is even harder to learn how to sacrifice for somebody one loves. But I am getting there. After all, without their encouragement, I would not have even returned to school.

Yes, I met so many nice people here at PENN. I thank Dr. Young-Suk Lee for her endless comments through email. A lot of her work and concepts were directly reflected in this paper. I thank Tilman Becker and Michael Niv for their comments especially on Chapter 5. I also thank Chang-Bong Lee for his thorough comments on linguistics. I thank the US military research officer, Mr. Yaeger for his kindness and help. I also thank Dr. Bonnie Dorr, Dr. Sungki Suh, and Jeyhoon Lee at the University of Maryland for their interest and cooperation. I cannot thank Mike Felker enough for his enormous help in many incidents. I also thank Angella Chung, Frances Lee, and Owen Binczewski for proofreading my thesis. I deeply thank Dr. In-Sup Lee, Jin-Young Choi, Dr. Martha Palmer, Dania Egedi, Young-Suk Lee and Jong-Cheol Park for their kindness and many invitations to "dinner".

Well, I did not realize it. But I have met so many nice people here at Philadelphia. As a matter of fact, I am already missing PENN.

[0]† This paper was sponsored by Battelle. Account No. 5-21630; Sponsoring No. 1326

Contents

List of Tables

List of Figures

Abstract

This paper addresses various issues related to representing the Korean language using Tree Adjoining Grammars. Topics covered include Korean grammar using TAGs, Machine Translation between Korean and English using Synchronous Tree Adjoining Grammars (STAGs), handling scrambling using Multi Component TAGs(MC-TAGs), and recovering empty arguments. The data for the parsing is from US military telecommunication messages.

Chapter 1

Overview

Linguistics and computer science are merging closer together in natural language processing. Especially in Korean, many scholars have made notable contributions including a wide spectrum of studies such as pragmatics of compound verbs, indirect speech acts, and parsing methods.

Unfortunately, most of them paid little attention to combining these ideas together to formalize linguistic theory into computer science, and vice versa. Beyond just observing a phenomenon, linguists should be able to formalize it, or give a reason to prove that their observation is right. Computer scientists should not just code programs, disregarding linguistic theories only to prove that their program is working in a small domain. For the researcher in the so called Computational Linguistics area, equal amounts of time should be devoted to both areas.

With that in mind, this paper aims to represent Korean grammar formally by using the formalism called Tree Adjoining Grammars[26]. As a computer scientist, I have to admit that my background in linguistics is very shallow. However, whenever I wanted to formalize Korean grammar, I tried to reflect current Korean linguistic theories as much as possible.

In doing so, I developed some of my approach based on different concepts from the current linguistic theory of Korean, as I felt that some of the theories were too syntax-oriented, and tried to explain everything within that boundary, not considering pragmatics or semantics. Actually, I believe that many phenomena in Korean (and other languages) involve semantics and pragmatics as well as syntax.

However, I also have to admit that the method or theory here is neither well documented, nor fully developed. All I can say is that I want to broaden the scope in analyzing some of the linguistic phenomena. Even though it is not well developed here, I explicitly state how I want to explain this phenomena from a different point of view in the future. In that respect, this thesis is a kind of proposal for the direction of my future study.

In Chapter 2, I briefly introduce the formalism of Tree Adjoining Grammars, which was first developed by Joshi, Levy, and Takahashi[26].

In Chapter 3, basic Korean Lexicalized Tree Adjoining Grammars are developed. I illustrate the Korean elementary trees, along with an explanation of the Korean grammar itself. The Korean grammar using TAGs was applied to the military domain messages shown in Appendix A. Some of the derived trees of the military messages are shown in Appendix B.

Chapter 4 deals with the actual translation issues using the Synchronous Tree Adjoining Grammar formalism, an extension of lexicalized, feature-based Tree Adjoining Grammars (FB-LTAGs). Here, I present how the basic idea of Synchronous TAGs can be used for Korean to English and English to Korean translation. To illustrate the coverage of the system and the effectiveness of semantic feature unification, various examples of syntactic phenomena are given such as relative clauses and wh-questions together with semantic phenomena such as accurate lexical selection for polysemous verbs. At the end of the chapter, Lexical Conceptual Structure in Synchronous Tree Adjoining Grammars is briefly discussed as an alternative approach to translation.

In US military telecommunication messages, I noticed two main phenomena that should be handled immediately for reasonable processing, as they are so prevalent. One was scrambling and the other, dropped arguments. Chapter 5 and 6 deal with these two phenomena.

In chapter 5, a computational method of handling scrambling using Multi-Component Tree Adjoining Grammars is presented. As HGs and CCGs are Mildly Context Sensitive in their generative capacity like TAGs, I include the work done by Young-Suk Lee and Michael Niv for Combinatory Categorial Grammars[35] and the work done by Ik-Hwan Lee for Head Grammars[32]. For interpreting scrambling phenomenon, my assumption is that scrambling is just random and it is not linguistic movement. Again, I am not in a position to argue that my assumption is correct. But I want to broaden the scope to explain scrambling as a phenomenon not restricted only by syntax but by semantics and pragmatics as well.

In chapter 6, I present a method of recovering empty arguments in Korean. The CCG code written in Prolog is listed in Appendix E for implementing the basic concept of the algorithm. However, the algorithm presented here is focused on semantic features only. I would like to augment the algorithm with other theories such as centering in the future.

Finally, several sections of chapter 4 have been published as "Korean To English Translation Using Synchronous TAGs" by Dania Egedi, Martha Palmer, Hyun S Park, and Aravind Joshi, in the Proceedings of the First Conference of the Association for Machine Translation in the Americas in Columbia, Maryland[14], and chapter 6 has been presented as "Recovering Empty Arguments in Korean" by Hyun S Park, Dania Egedi, and Martha Palmer, at the 1994 Joint Conference of 8th Asian Conference on Language, Information, and Computation and the 2nd Pacific Asia Conference on Formal and Computational Linguistics, in Kyoto, Japan,[41], and these papers were mildly modified.

Chapter 2

The Formalism of TAGs

Tree Adjoining Grammars (TAGs) were first developed by Joshi, Levy, and Takahashi[26]. As first shown by Joshi and Kroch, the properties of TAGs permit us to encapsulate diverse syntactic phenomena such as unbounded dependencies in a natural way[25][29].

A Tree-Adjoining Grammar consists of a quintuple (Σ, NT, I, A, S), where Σ is a finite set of terminal symbols, NT is a finite set of non-terminal symbols, S is a distinguished non-terminal symbol, I is a finite set of finite trees, called initial trees, and A is a finite set of finite trees, called auxiliary trees[55].

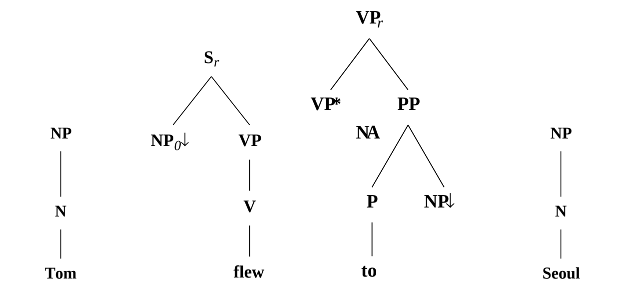
Later, Yves Schabes, Anne Abeille, and Aravind K. Joshi extended Tree Adjoining Grammars to include lexicalization[49]. Lexicalized grammars systematically associate each elementary structure with a lexical anchor. The *grammar* consists of a lexicon where each lexical item is associated with a finite number of structures for which that item is the anchor (denoted with the $\diamond$ symbol next to the node name).

A TAG is a tree-rewriting system and TAGs generate phrase-structure trees. There are no separate grammar rules, although there are combining rules for combining these structures, i.e., *adjunction* and *substitution*.

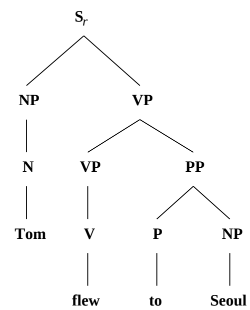
2.1 Elementary Trees

There are two kinds of elementary trees in TAGs: initial trees and auxiliary trees.

In describing natural language, initial trees are minimal linguistic structures that contain



(a) Tom (b) Flew (c) To (d) Seoul



(e) Derived Tree

Figure 2.1: Lexicalized Elementary Trees and a Derived Tree

no recursion. An initial tree is called an S-type tree if its root is labeled with type S. In initial trees, all internal nodes are labeled by non-terminals and all leaf nodes are labeled by terminals or by non-terminal nodes marked for substitution. Trees 2.1(a), 2.1(b), and 2.1(d) are initial trees. By convention, initial trees are called an α tree.

Recursive structures are represented by auxiliary trees, which represent constituents that are adjuncts to basic structures. In auxiliary trees, all internal nodes are labeled by non-terminals and all leaf nodes are labeled by terminals or by non-terminal nodes marked for substitution, except for exactly one non-terminal node, called the foot node. The foot node has the same label as the root node of the tree. Figure 2.1(c) is an auxiliary tree. By convention, auxiliary trees are sometimes called a β tree.

A down arrow ($\downarrow$) is used with nodes to mark a substitution node, and an asterisk ($*$) is used with nodes to mark a foot node.

Figure 2.1(e) shows the derived tree for *Tom flew to Seoul* built from elementary trees, Figure 2.1(a), (b), (c), and (d). As was discussed before, there are no grammar rules. So, to combine each elementary tree to get the final derived tree in Figure 2.1(e), we need universal combining rules.

2.2 Two Operations in TAGs

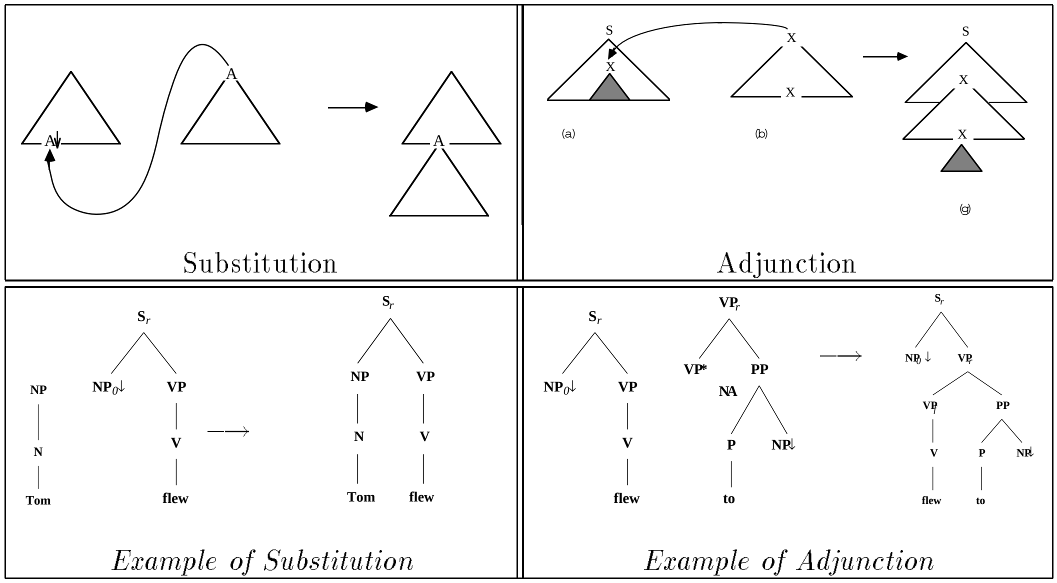


Figure 2.2: Combining Operations

As there are no grammar rules in TAGs, combining operations are needed to combine each lexicalized structure. There are two operations defined in Tree Adjoining Grammars, namely, *substitution* and *adjunction*.

Substitution can take place only on non-terminal nodes of the frontier of the tree, and a substitution node is marked by a down arrow (↓). In the *substitution* operation, a node marked for substitution in an elementary tree is replaced by another elementary tree whose root label is the same as the non-terminal. So, in Figure 2.2, A↓ is replaced by the tree on the right side, whose root label is A.

In an *adjunction* operation, an auxiliary tree is inserted into an initial tree. The root and foot nodes of the auxiliary tree must match the node label at which the auxiliary tree adjoins. Actually, it is this operation that makes lexicalization possible. Technically, *substitution* is only a specialized version of *adjunction*[1]. The adjunction operation is shown on the right of Figure 2.2.

[1]For more discussion about lexicalization, see "XTAG User Manual version 1.0" [50].

2.3 Feature-Based LTAGs

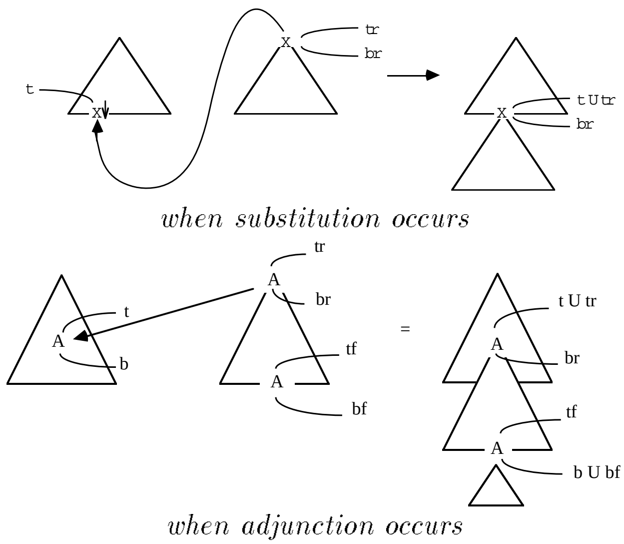


Figure 2.3: Updating of Feature Structures

The Feature-Based Lexicalized Tree Adjoining Grammar formalism (FB-LTAG) is based on the Tree Adjoining Grammar[26], which has been extended to include lexicalization [49][48], and unification-based feature structures[56].

Each node of an elementary tree is associated with two feature structures, the top and the bottom. The bottom feature structure contains information relating to the subtree rooted at the node, and the top feature structure contains information relating to the supertree at that node. For example, node

V [**mode:** <1> **ind**]
[**mode:** <1>
arg : [**num:** **singular**, **pers :** **3**]]

has three paths: <arg num>, <arg pers>, and <mode> in the bottom feature structure. The **mode** in the top feature will be unified with **mode** in the bottom feature, and it was indicated by the same variable, <1>. Substitution nodes have only a top feature structure, while other nodes have both a top and bottom one.

Figure 2.3 shows an auxiliary tree and an elementary tree, and the tree resulting from a *substitution* operation and an *adjunction* operation.

In the *substitution* operation, the features of the node at the substitution site are the unified features of the original nodes. The top feature structure of the node is the result of unification of the top features of the two original nodes, while the bottom feature structure of the new node is simply the bottom features of the root node of the substituting tree. So, in Figure 2.3, the top feature structure, t of X$\downarrow$ should unify with the top feature structure, tr of the root node X.

In the *adjunction* operation, the top feature structure of nonterminal node, $\mathbf{A}$ unifies with the top feature structure of the root node of the auxiliary tree, $\mathbf{A}$, while its bottom feature structure unifies with the bottom feature structure of the foot node, $\mathbf{A}$ of the auxiliary tree, in the right side of Figure 2.3.

Lexicalized trees, such as those seen in Figure 2.1, allow individual lexical items to instantiate the feature structures in the trees with lexically specific information. This may include, for instance, constraints that verbs place on their complements, or morphological and semantic information associated with an individual word. In lexicalized TAGs, at least one terminal symbol (the anchor) must appear at the frontier of all initial or auxiliary trees.

Nodes of elementary trees may specify constraints on the set of auxiliary trees that can adjoin to them. These constraints enforce obligatory adjunction of any auxiliary tree, selective adjunction of a specified set of auxiliary trees, or no adjunction at all. Throughout the paper, Selective Adjunction is represented as $\mathcal{SA}$, Obligatory Adjunction as $\mathcal{OA}$, and Null Adjunction as $\mathcal{NA}$[2].

2.4 Summary

The formalism of FB-LTAGs is introduced, here. There are other variants of TAGs such as Synchronous Tree Adjoining Grammars (STAGs)[51], and Multi-Component Tree Adjoining Grammars (MC-TAGs)[6][3]. STAGs is used for machine translation, and will be discussed in chapter 4, and MC-TAGs is used for handling scrambling, and will be discussed in chapter 5.

[2] Vijay and Joshi have shown that the obligatory and selective adjunction constraints can be simulated using linguistically motivated features on the node[56].

[3] The concept of *Super Parts of Speech* (Supertags) is implemented for the current XTAG system by Joshi and Srinivas[27].

Chapter 3

Korean Grammar In TAGs

How TAGs can be employed for representing Korean Grammar will be explored here, together with Korean Grammar and some current linguistic issues.

Lexical items are defined by the tree structure or the set of tree structures they anchor. An anchor node is specified by a node name with a $\diamond$ sign. In the current TAG system, when a word can have several structures, it is treated as several lexical items with different entries in the Syntactic Lexicon. Words are marked with the appropriate morphological features in the Morphological Lexicon[1]. The morphological lexicon associates a word with an abstract class of words, a preterminal symbol, and a set of morphological features. Sentential clauses are considered as elementary trees, usually anchored by their main verb, sometimes together with other clausal complementizers, or suffixes.

[0]† The abbreviations used in this paper are as follows:

- For Linguistic Terms :

NOM : nominative	ACC: accusative	DAT : dative
TOP : topic	CE : causative ending	PRES : present
Pl : plural	PASS : passive	DECL : declarative
NEG : negative	PAST : past	COMP : complementizer

- Nodes for Korean TAGs:

SP : Subject Phrase (NP + Nom)	OP : Object Phrase (NP + Acc)
S : Sentence	SFX : Suffix (Conjugational Form for Verb)
VP : Verb Phrase	NP : Noun Phrase

[0]‡ The terminology of Korean grammar follows the book, "Korean Grammar for International Learners"[21].

[1]The "Syntactic Lexicon" and "Morphological Lexicon" here are the terms used in XTAG, and should not be confused with linguistic terms.

3.1 Korean Alphabet

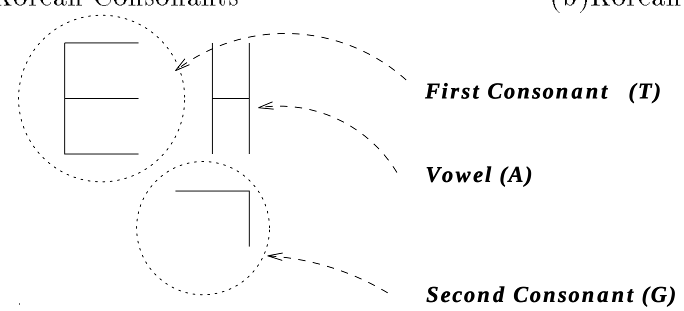
Korean has 19 consonants, 10 simple vowels and several compound vowels.[2] Korean characters are syllable-oriented; each Korean syllable can be represented by a first consonant group and one vowel with an optional second consonant. The optional second consonant group appears at the bottom of a vowel. Table 3.1(a) and (b) shows the actual Korean characters together with the *Yale Romanization System*[3]. Figure 3.1(c) shows the actual character 택 (TAG). To make a character, 택 (TAG), A consonant ㅌ[t] together with the vowel ㅐ[æ] makes a sound [tæ]. Then, an optional second consonant ㄱ[g] is added at the bottom of the vowel ㅐ[æ], making it sound like [tæg][4]. Thus, the syllable structure of Korean is square as opposed to the linear one of English.

ㄱ(k)	ㄴ(n)	ㄷ(t)	ㄹ(l)	ㅁ(m)
ㅂ(p)	ㅅ(s)	ㅇ(-ng)	ㅈ(c)	ㅊ(ch)
ㅋ(kh)	ㅌ(th)	ㅍ(ph)	ㅎ(h)	
ㄲ(kk)	ㄸ(tt)	ㅃ(pp)	ㅆ(ss)	ㅉ(cc)

(a)Korean Consonants

ㅏ(a)	ㅑ(ya)	ㅓ(e)	ㅕ(ye)
ㅗ(o)	ㅛ(yo)	ㅜ(wu)	ㅠ(yu)
ㅡ(u)	ㅣ(i)	ㅐ(ay)	ㅔ(ey)

(b)Korean Vowels



(c) TAG in Korean

Table 3.1: Korean Consonants and Vowels with TAG(택) example

3.2 Sentence Structure

The basic sentence types in Korean can be represented by simple combinations of a subject and a predicate. There are four basic sentence types in Korean, which are determined by the

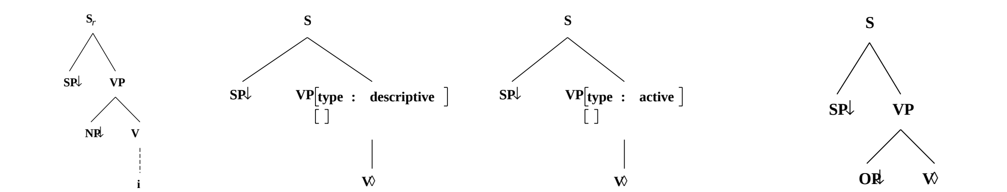
[2]The other complex vowels, not shown in the table, are 와 (wa), 왜 (way), 외 (oy), 얘 (yay), 예 (yey), 위 (wi), 웨 (wey), 워 (we), 의 (uy).

[3]The *Yale Romanization System* is somewhat different from what native Korean speakers are likely to understand. But this is the way Korean characters can be expressed in alphabetical form

[4]The actual notation in *Yale Romanization System* for the word 택 (TAG) is *thayk*

type of the verb used in the sentence[21]. Figure 3.1 shows the four basic sentence types. Figure 3.1(b) and Figure 3.1(c) can be distinguished only by their semantic features. Figure 3.1(a) is the *ita* verb tree.

Ita (이다) is a Korean version of the copula verb *be* in English, which shows the equivalent relationship between the subject and the predicative noun or the inclusive relationship of the subject within the predicative noun. Usually a noun must be placed just before *ita*. Figure 3.1(b) describes the state or the characteristics of a thing, whereas Fig 3.1(c) describes the movement or the action of a thing. Figure 3.1(d) is similar to the transitive verb construction in English except that the order is SOV. Each S-type sentential clause tree is anchored to the verb stem node V◇.



(a) S + N ita 주어+서술동사 (b) S + V_{desc} 주어+형용동사 (c) S + V_{act} 주어+동작자동사 (d) S + O + V 주어+목적어+동작자동사

Figure 3.1: Four Kinds of korean S tree

In the basic structure, there exists only one verb. Nouns are usually followed by particles which function as auxiliaries. These particles mark the case of a noun. For example, -*i* or -*ka* are attached after nouns which are the subjects of the sentence, while -*ul* or -*lul* is attached after nouns which are the objects of the sentence. The four structures in Fig 3.1 show the canonical order of each category[5].

Two sentences or more may join together to form a complex sentence. When sentences are combined together, a conjunctive ending is added to the verb stem, depending upon the intended message. In sentence (1), the verb stem *hayngpokha* takes -*myen* as its conjugational form[6]. *Myen* roughly functions like *if* in English. However, unlike English, *myen* follows the verb stem. So, in sentence (1), when the speaker starts with *Jerry-ka hayngpokha*, there is

[5]It is worth mentioning that Korean allows considerable freedom in word order; the only strict restriction is its verb-final property. More issues related with this phenomenon, called scrambling, will be dealt in Chapter 5.
[6]Further conjugational forms for conjunctive structure are shown in Table 3.2.

no way to tell whether it will be just a simple sentence or a complex sentence. If *ta* follows it, then it just means *Jerry is happy* and the sentence ends there. However, if *myen* or any conjunctive suffix follows it, then as the meaning will be *if Jerry is happy*, an additional sentence is expected. Figure 3.2 shows the TAG notation for sentence (1). Unlike the sentential clause structures in Figure 3.1, the conjunctive structure is anchored to the node SFX◇ with the node feature, [suffixtype: conjunctive]. In other words, it is implemented as an adjunction(β) tree. The elementary tree is represented with a bold line in Figure 3.2

	Jerry-ka	hayngpokha-myen,	Tom-i	pwulkoyhata.
1	제리가	행복하면,	톰이	불쾌하다
	Jerry-NOM	happy-myen,	Tom-NOM	unhappy-ita.
	If Jerry	is happy,	Tom	is unhappy.

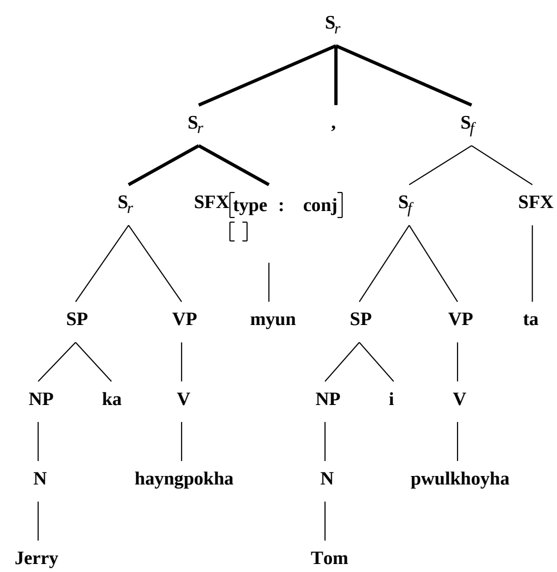


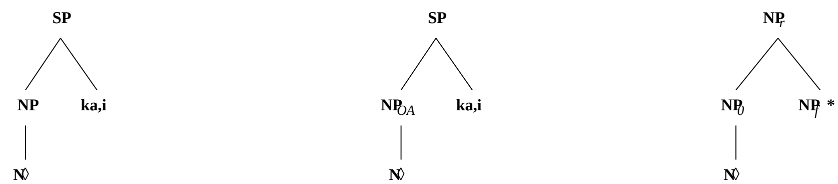
Figure 3.2: Conjunctive β Tree

3.3 Nouns and Pronouns

Nouns are divided into independent nouns and dependent nouns. Independent nouns are further divided into common nouns and proper nouns. The case of nouns is assigned, by using case particles. Generally, nouns can be made plural by adding the suffix *dul* (들). There is no gender differentiation of nouns in Korean. There are no definite or indefinite articles, either.

Pronouns in Korean may be divided into three types : Personal Pronouns, Demonstrative Pronouns, and Interrogative Pronouns[7]. Figure 3.3(b) is a tree for a dependent noun. The only difference from Figure 3.3(a) is that it has an $\mathcal{OA}$ constraint on node NP, meaning that the dependent noun must be adjoined by another NP-type tree.

Wh words have a feature [wh:+]; wh words do not force movement, unlike English.



(a) Independent Noun Tree (b) Dependent Noun Tree (c) Tree for Collocation

Figure 3.3: Trees for Subject NOUNs

3.4 Particles

There are various particles which show whether the elements preceding them are classified as subject, object, dative, etc. Particles are classified into three categories depending upon their functions : the case particle, the auxiliary particle, and the connective particle. The case particle follows a nominal and determines its case. Its main usage is to manifest the grammatical functions of its host nominals within a sentence.

The case particle is further classified into three types : the nominative particle which follows a subject, the objective particle which follows an object, and the adverbial particle which follows an adverb.

The auxiliary particles add a special meaning to a word. They may be attached after a noun, another particle, an adverb, or even a verb. Some of them are sometimes called topic markers. Topic Particle or auxiliary particle *nun/un* (는/은) is used to contrast something or to simply present a topic. *Un* (은) follows a consonant and *nun* (는) follows a vowel. When used

[7]Korean does not have a relative pronoun.

[7]†Particle types in Table :

Nominative	-i,-ka
Objective	-ul,-lul
Adverbial	-eykey,-ulo
Auxiliary	-un,-nun
Conjunctive	-wa,-kwa
Adnominal	-uy

with nouns which become the subject or the object of the sentence, *nun/un* (은/는) replaces the nominative and/or objective particle. This replacement of another particle commonly takes place with the auxiliary particle.

The connective particles connnect a noun to a noun. They can be either a conjunctive particle or an adnominal particle. If the preceding noun ends with a consonant, *kwa* (과) is used; otherwise, *wa* (와) is employed. Fig 3.4 shows each type of particle tree. Whereas subject and object particles are represented as substitution (α) trees, adnominal, adverbial, and connective particles are represented as adjunction (β) trees.

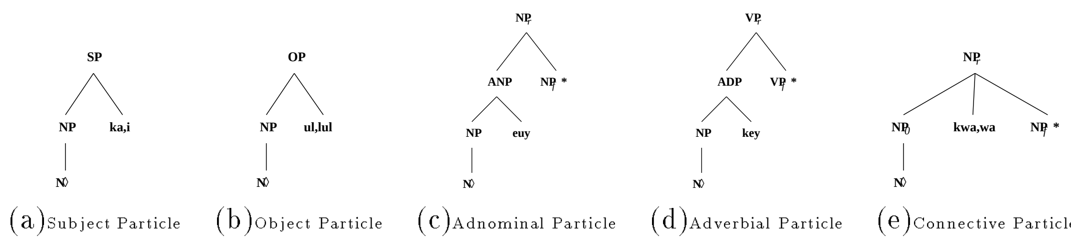


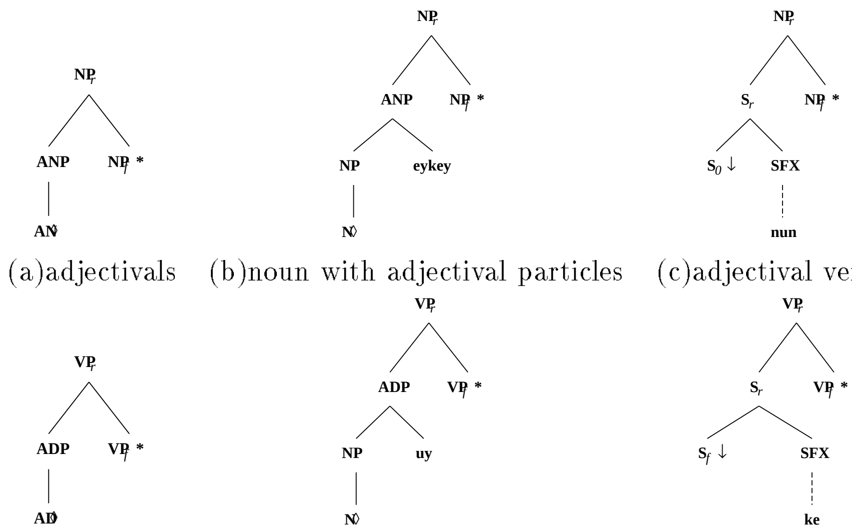
Figure 3.4: Particle Trees

3.5 Adnominals & Adverbials

Any word in the basic sentence structure can be modified by its modifiers. Modifiers are either *adjectivals* (형용사) or *adverbials* (부사). *Adjectivals* modify nouns, whereas *adverbials* modify verbs or sentences. *Adjectivals* include adjectives, nouns with adjective particles, and adjectival verbs. *Adverbials* include adverbs, nouns with adverbial particles, and adverbial verbs. Figure 3.5 shows the modifier structures. The particles shown in Figure 3.5 are just one of the examples of the possible particles of that category. Notice that Figure 3.5(c) and (f)'s anchor node is SFX.

Adnominals modify the nouns which follow them. There are not many pure *adnominals*; a lot of *adnominals* are just one conjugative form of verbs. Adverbials can modify other adverbials or verbs.

There is no syntactic difference between interrogative and other type of *adverbials*, which implies that there is no wh movement in Korean. For example, Korean has an interrogative adverbial which is similar to the meaning of *which* in English. However, it does not force the wh word to move to some other place.



(a)adjectivals (b)noun with adjectival particles (c)adjectival verbs

(d)adverbs (e)noun with adverbial particles (f)adverbial verbs

Figure 3.5: Modifiers

3.6 Verb

Various meanings are given to verbs by conjugation. The verbal structure in Korean consists of a *verb stem* plus *suffixes*. The stem stays the same, while the suffix may be conjugated. Suffixes differ as to whether they combine with the stem to form a word, or form another stem, to which some additional suffix must be added. Cho divided Korean verb suffixes into three classes[9].

- Stem-forming suffixes combine with a stem to form a complex stem. A verbal word may contain one or more stem-forming suffixes.
- Word-forming suffixes combine with a stem to form a word. A verbal word must contain exactly one word-forming suffix. Also, the word forming suffixes must be final.
- Word suffixes combine with a word to form a word.

Suffixes	Class	Meaning
ka-ta(가다)	Word-forming	Original
ka-nta(가-ㄴ다)	Word-forming	Declarative
ka-ni(가-니)	Word-forming	Interrogative
ka-pnika(가-ㅂ니까)	Word-forming	Interrogative
ka-myen(가-면)	Word-forming	Connective
ka-nun(가-는)	Word-forming	Adnominal
ka-m(가-ㅁ)	Word-forming	Nominal
ka-si(가-시)	Stem-forming	Honorific
ka-ss(가-ㅆ)	Stem-forming	Past
ka-ko(issta)(가-고 (있다))	Stem-forming	Progressive
ka-keyss(가-겠)	Stem-forming	Future
ka-ssessess(가-ㅆ었었)	Stem-forming	Past-Participle
ka-myen(가-면)	Word-forming	tensed-COMP
ka-ko(가-고)	Word-forming	quotative-COMP
ka-yo(가-요)	Word-forming	discourse-Suffix

Table 3.2: *Ka* Verb and its Conjugational Forms

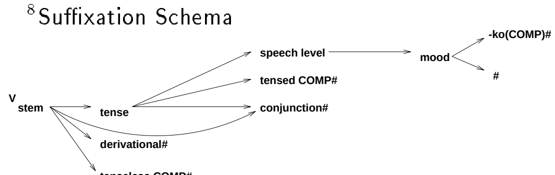
Thus, every verbal word in Korean is of the form:

V_{stem} + Stem-forming suffix* + Word-forming suffix[8]

In Korean, morphological variations of a verb change the structure of the verb phrase as a whole, whereas the verb stem always keeps the same structure. For this reason, whereas a stem-forming suffix[9] together with a verb stem are treated as one syntactic item with different morphological forms, word-forming suffixes are implemented as separate syntactic items[9] [10]. In the case of irregular verbs, all the morphological forms are put in the Morphological Lexicon.

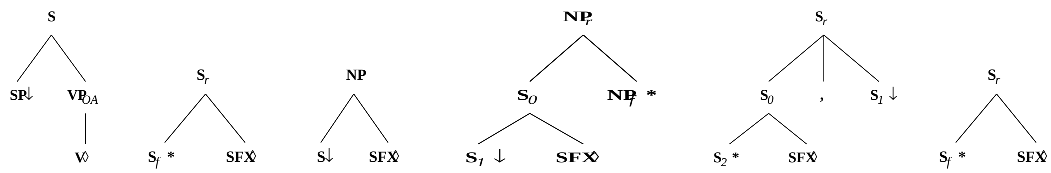
Table 3.2 summarizes the verb conjugation rule for the verb stem *ka* (가) or *go*. The original morphological form of the verb is *Verb Stem* + ta. Unlike English, there is no Subject-Auxiliary Inversion in Korean: an interrogative sentence is represented by a conjugational form -*ni* or -*pnika*.

[8] Suffixation Schema



[9] In TAGs, the node for the verb suffixes is named SFX◇.

[10] In English, the actual grammar consists of a morphological lexicon, which lists the possible morphological variations for a word, and a syntactic lexicon.



(a)ka (b)ka-ta (c)ka-m (d)ka-nun (e)ka-myen (f)ka-ni

Figure 3.6: *Ka* Verb Trees

Tenses in Korean are determined from the speaker's point of view. Sometimes, tenses communicate certain aspects of the verb actions or states rather than the time reference of the verb[11]. Figure 3.6 shows the ka verb structures. Figure 3.6(a) is the original stem tree, whose anchor node is V$\diamond$. This tree has an $\mathcal{OA}$ constraint on a VP node, meaning that the Word-Forming Stem must be adjoined to generate a grammatical structure. Figure 3.6(b) is the declarative conjugational form. Figure 3.6(e) is the conjunctive structure. Figure 3.6(f) is the interrogative conjugational form. Unlike English where an interrogative form takes a subject-verb switch, Figure 3.6(b) and Figure 3.6(f) can be distinguished only by features, even though the structure is identical. Conjugational endings not only change the meaning of verbs, but also the case of the verbs in the sentence. For example, *ka-nun (가는)* is an adnominal, which is just one of the conjugational forms of the verb stem ka.

Figure 3.6(d) shows the structure of an adnominal verb conjugation structure. Actually, it takes the form of a relative clause with the subject or object missing. In other words, the sentence structure that will be substituted into the $S_1\downarrow$ node should be a sentence with one of its arguments missing (this can be achieved by imposing a constraint on the $SP_1\downarrow$ node). This tree has an anchor node SFX, and will be lexicalized with nun suffix. Also, notice that this tree is a β tree (adjunction tree), as it is modifying a noun phrase NP. Figure 3.6(c) shows another example of verb conjugation changing into another case; if the verb stem ka takes a conjugational form m, it becomes a noun *ka-m (감)*. S$\downarrow$ will have a feature restriction so that only the Figure 3.6(a) type S tree can be substituted. Stem Forming Suffix was implemented and put into the Morphological Lexicon. So, the syntactic entry ka will have a V as its category, and morphological entries such as *ka-kess (가겠) : [tense : future]* or *ka-ss (갔) : [tense : past]*.

[11]Suffixes such as i (이) or hi (히) can be used for changing a verb into passive morphological form, which is not shown in Table 3.2 as the passivization of the verb *ka (가)* is semantically impossible

Auxiliary verbs follow main verbs and assist the meaning of the main verbs, which is the other way around in English. When auxiliary verbs are combined with main verbs, special conjunctive endings for the main verb (*-e, -ko, -ci, -ke*) are used. Most of the auxiliary verbs are used as a main verb, also. Exceptions are *cita* and *sipta* which cannot be used as a main verb.

3.7 Negation

There are two types of negative sentences in Korean. They have been differentiated as *verb* vs. *sentence* negation, or *simplex* vs. *complex* negation. For the sentence *Jerry-ka talinta* or *Jerry runs*, two types of negations are possible.

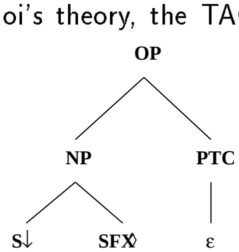
	Jerry-ka	talinta.		
2 [Aff-V]	제리가	달린다		
	Jerry-Nom	run		

	Jerry-ka	an	talinta.	
3 [Neg-V]	제리가	안	달린다	
	Jerry-Nom	not	run	

	Jerry-ka	talici	ani	hanta.
4 [Neg-S]	제리가	달리지	아니	한다.
	Jerry-Nom	running-(ACC)	NEG	do

Choi claims that even though the semantic distinctions cannot be easily captured, there is a semantic disparity between the two types of negative sentences in Korean[10]. The affirmative corresponding to sentence (3) is (2). The affirmative corresponding to (4) is shown in (5).

	Jerry-ka	taliki-lul	hanta
5 [Aff-S]	제리가	달리기를	한다
	Jerry-Nom	running-ACC	do

In other words, *Jerry-ka taliki* has become a noun clause. By adding the ACC particle *lul*, *hanta* or *do* now has a whole sentence as an object. The negation form is shown in sentence (4). Following Choi's theory, the TAG tree for the negation form *ci* will be represented as OP-type structure, 

, where the anchor node is still SFX.

3.8 Narratives

Narrative refers to quoting one's speech or written words at a later time[12]. The basic word order of narrative is 'Subject + Addressee + Quoted Sentence + Quotation Particle + Predicate'. Here, the subject is the original speaker of the quoted sentence, and the addressee is the original hearer. Depending upon the type of quoted sentence, special quotation verbs are used as the predicate.

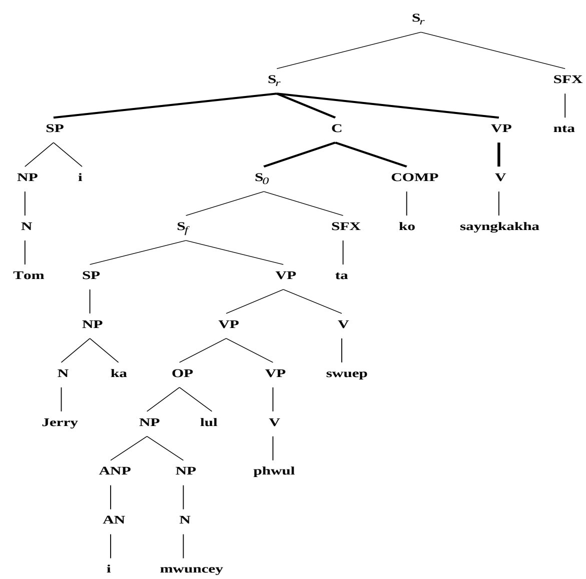


Figure 3.7: Narrative Structure

For example, *sayngkakha-ta* (생각하다 or think) is used when the quoted sentence is declarative, *mwut-ta* (묻다 or ask) when the quoted sentence is interrogative, *myenglyunghata* (명령하다 or order) when the quoted sentence is imperative, and *cyeanhata* (제안하다 or suggest) when the quoted sentence is propositive[21]. The use of indirect quotation in Korean is accomplished by a combination of a quoted sentence, a quotation particle *-ko*, and a quotation verb. The relationship between the quoted sentence and the quotation verb, which determines the sentence meaning, is manifested in different ways according to the various types of quotation verbs used.

[12] The term 'quote' here is slightly different from the literal meaning of 'quote' in English.

	Tom-i	[Jerry-ka	i mwunce-lul	pwul swu	epta]-ko	sayngkakhanta.
6	Tom-NOM	Jerry-NOM	this problem-ACC	solve	cannot-ko	think.
	톰이	제리가	이 문제를	풀 수	없다고	생각한다.
	Tom	thinks	that	Jerry	cannot	solve this problem.

Figure 3.7 shows the derived tree of the sentence (6). The elementary tree which is noted with a bold line has two anchor nodes: COMP$\diamond$ lexicalized with *ko* and V$\diamond$ lexicalized with *sayngkakha*.

3.9 Causative structure

The causative structure in Korean has received many different views from many different schools of linguists, as this construction has both monoclausal and biclausal properties.

Korean has several causative particles, one of which is -*key*. The other form is the lexical causative, which is formed by infixation of *i, hi, gi, li* to verbs or adjectives. The sentences (7), (8), and (9) are examples of Korean sentences of causatives, meaning *Tom caused Jerry to go*.

	Tom-i	Jerry-ka	ka-key	haytta.
7	Tom-Nom	Jerry-Nom	go-CE	made.
	톰이	제리가	가게	했다.

	Tom-i	Jerry-lul	ka-key	haytta.
8	Tom-Nom	Jerry-ACC	go-CE	made.
	톰이	제리를	가게	했다.

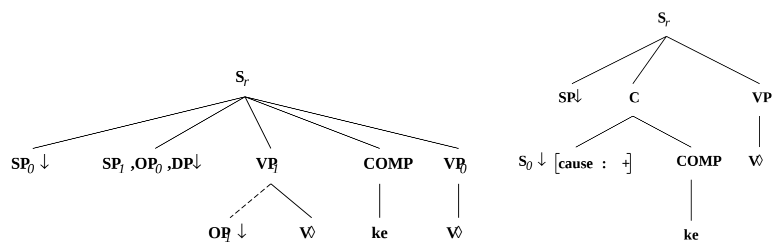
	Tom-i	Jerry-eykey	ka-key	haytta.
9	Tom-Nom	Jerry-DAT	go-CE	made.
	톰이	제리에게	가게	했다.

According to Young-Suk Lee, the causee in a Korean causative sentence may be marked with the nominative, accusative or dative marker as long as it has an [animate:+] feature[33]. In sentence (7), the causation is permissive, whereas in (8), it is coercive. She argues that the causee may be marked with the nominative, accusative, or dative case markers except when the causee is [animate:-], in which case, it should be marked as either nominative or accusative, but not as dative. For *Jerry made the kitchen clean*, sentence (12) is not grammatical, whereas sentence (10) or sentence (11) is grammatical.

	Jerry-ka	puek-ul	kkaykkussha-ke	ha-yet-ta.
10	Jerry-NOM	kitchen-ACC	clean-CE	cause-PAST-DEC.
	제리가	부엌을	깨끗하게	하였다

	Jerry-ka	puek-i	kkaykkussha-ke	ha-yet-ta.
11	Jerry-NOM	kitchen-NOM	clean-CE	cause-PAST-DEC.
	제리가	부엌이	깨끗하게	하였다

	*Jerry-ka	puek-eykey	kkaykkussha-ke	ha-yet-ta.
12	Jerry-NOM	kitchen-DAT	clean-CE	cause-PAST-DEC.
	제리가	부엌에게	깨끗하게	하였다



(a) mono-clausal (b) bi-clausal

Figure 3.8: Monoclausal & Biclausal Analysis of Causative Structure

In the following script from [Lee, 89], she compares causative structure with narrative structure by analyzing the scope of the adverbial for each structure.

"... In accounting for the causative construction, the primary concern has been focused on the case variation of the causee. Most of the analysis gave an explanation that the dative or the accusative case-marked causee is an argument of a simplex clause, which is derived from a complex structure with an embedded clause through reanalysis. This analysis assumes the following: The causative construction is biclausal at D-structure, which is responsible for the nominative causative. Reanalysis, which combines the matrix verb *hata* with the embedded verb to form a complex verbal, results in a simplex structure. If the embedded verb is intransitive, the causee becomes the accusative argument, whereas if the embedded verb is transitive, the causee becomes the dative argument.

This analysis has a problem explaining a wider range of data and shows some phenomena which reveal that causative structure is different from a simplex sentence. Depending on the transitivity of the embedded verb, it allows the causee to be marked with either the dative or accusative case but not both. However, it can be seen that the causee can be marked both accusative and dative regardless of the transitivity of the embedded verb.

In sentence (13), the scope of adverbial *ppalli* or *quickly* is ambiguous introducing two interpretations. However, in sentence (14), such an ambiguity does not exist. There is a biclausal property which separates -*key*, the causative ending with -*ko*, the narrative complement..."

	Tom-i	Jerry-lul	ppalli	phisinha-key	ha-yet-ta.
13	Tom-NOM	Jerry-ACC	quickly	escape-CE	cause-PAST-DEC.
	톰이	제리를	빨리	피신하게	하였다
	"Tom quickly	had Jerry escape."	or	"Tom had Jerry	escape quickly."

	Tom-i	Jerry-lul	ppalli	phisinsikiessta.
14	Tom-NOM	Jerry-ACC	quickly	escape-CS-PAST-DEC.
	톰이	제리를	빨리	피신시키었다
	"Tom quickly had	Jerry escape."		

With monoclausal analysis, the TAG trees can be represented as in Figure 3.8(a) for the S(O)V structure with two anchor nodes of verbs. The biclausal causative structure is shown in Figure 3.8(b). Further study of causative structure is necessary to give more precise TAG representation.

Most recently, Bratt argues that the Korean periphrastic causative with an accusative causee is monoclausal, while the causative with a nominative causee is biclausal[8]. He presents three kinds of evidence: scrambling, negative scope, and adverbial case marking.

3.10 Multiple Subject Constructions

There is a sentence construction called *Multiple Subject Construction*. Literally, it means that a verb has more than one subject as its argument, or rather, it has more than one NP in nominative case.

	Tom-i	tali-ka	apu-ta.
15	Tom-Nom	leg-Nom	hurt-DECL.
	톰이	다리가	아프다

Some studies approach the multiple subject construction in terms of the subject-predicate relation[40], while others argue that the subject NPs, except for the rightmost one, are transformally introduced[30]. Yang tried to account for this double subject construction in terms of case grammar[62].

To account for sentence (15), Kuno attempted to explain it in terms of subjectivization transformation[30]. According to this theory, the first NP, *Tom-i* is derived from a possessive

NP, thus allowing the change of categorial status. This approach has been rejected because of the entailment of the change of categorial status. Another approach has attempted to account for it in terms of the base-generated subject-predicate relation[40]. In this case, *Tom-i* is the subject of the whole sentence, and the sub-sentence *tali-ka aputa* is the predicate of the whole sentence, giving the analysis of [$_S$ Tom-i [$_s$ tali-ka aputa]]. This approach was not favored as the explanation was not given in syntactic terms.

Syntactically, what multiple subject construction is saying is that Korean allows two subjects in one clause, whereas it is simply ungrammatical in English. Semantically, however, both the predicate *aputa* in Korean and *hurt* in English require Experiencers and Locations as their arguments; only in Korean, both arguments are provided in the form of a subject. In English, the Experiencer for the verb *hurt* usually has the form of a possessive NP (such as 'my' in 'My leg hurts.'). In Korean, however, it is possible to say *'tali-ka aputa'*, without specifying the Experiencer. However, even in this case, the Experiencer should be recoverable from the context. Unless, it is ungrammatical.

From such a view point, *i* of *Tom-i* in sentence (15) is just functioning as a topic marker[13]. This does not imply that only the first subject should be a topic. It depends on the sentence and the context.

16	Tom-i	i mwuncey-ka	phwul swu-epta ko	sayngkakhanta.
	Tom-NOM	this problem-NOM?	solve-NEG-COMP	think-PRES-DEC
	탐이	이 문제가	풀수 없다고	생각한다
	Tom_i thinks that	he_i can not	solve this problem.	

Sentence (16) has two subjects, *Tom-i* and *mwuncey-ka*. However, *ka* in *mwuncey-ka* is functioning as a topic particle, too, even though the eventual function of *ka* particle here is accusative. In case of Korean, there seems to be some some kind of overlapping of cases as well as overlapping of the function of the case.

This gives a good reason why we should pay attention to semantics as well as syntax. Furthermore, verbs related to this kind of construction have characteristics that can be distinguishable from other verbs. In other words, the clause should be either equative, intransitive, or passive, and these predicates are usually stative and do not allow an agent who acts[62].

[13] Also, 'Tom-i aputa' is grammatical. However, I think it is a problem of subcategorization. In other words, 'aputa' here has a different sense with aputa in sentence (15).

If these constructions are to be represented in the TAG formalization, we can set some features such as [2subject:+] for each predicate and can have two or three subject substitution nodes SP↓ in an elementary tree. Even though just including this kind of structure in TAGs is quite simple, much attention needs to be drawn to a semantic rather than a purely syntactic explanation of this phenomenon.

Levin has pointed out that the case system adopts arbitrary categorizations because it cannot represent the complexity of events, and case names encode semantic concepts without explicitly defining their properties and interaction. Case itself neither specifies the semantic role nor is it consistent.

Trying to explain the multiple subject construction in terms of syntax is problematic as the case itself is functioning in a different way. Actually, the title "Multiple Subject Construction" itself is misleading as I see this as a problem of case overlapping. Much attention needs to be focused on the role of the case marker.

3.11 An Example

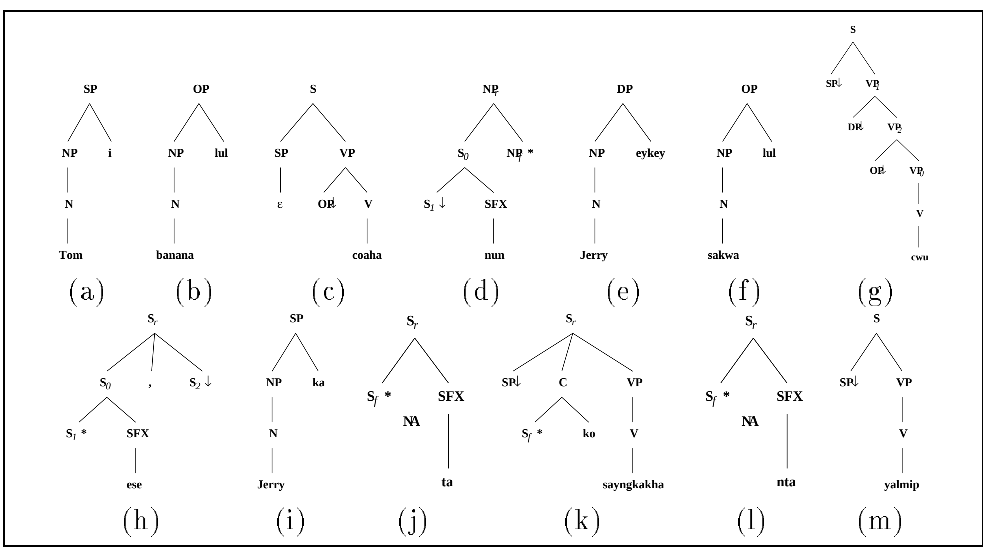


Figure 3.9: Elementary Trees For an Example

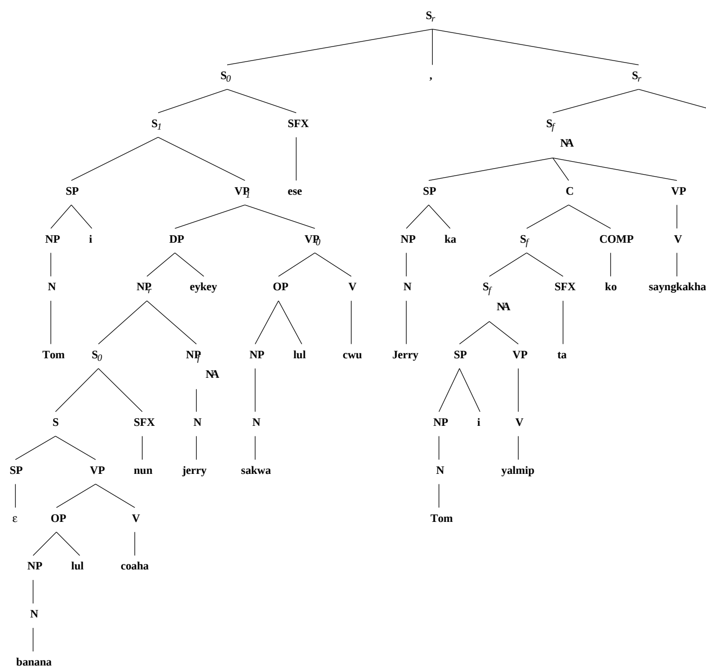
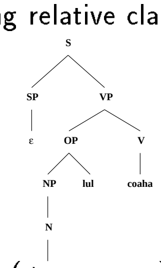


Figure 3.10: Final Derived Tree

17	Tom-i	Banana-lul	coaha-nun	Jerry-eykey	sakwa-lul	cwu-ese,
		Jerry-ka	Tom-i	yalmipta-ko	sayngkakha-nta.	
	Tom-NOM	Banana-ACC	like-NUN	Jerry-ACC	apple-ACC	give-CNJ,
		Jerry-NOM	Tom-NOM	bad-COMP	think-PAST-DECL.	
	톰이	바나나를	좋아하는	제리에게	사과를	주어서
		제리가	톰이	얄밉다고	생각한다	
	Since	Tom gave	an apple to	Jerry	who loves bananas,	
		Jerry	thinks	Tom	is mean.	

Now, with the elementary trees, we can parse sentence (17) , which has a relative clause, a narrative structure , and a conjunctive structure. Figure 3.9(c) is an elementary tree lexicalized with *coaha*. This tree represents an elementary tree for a subject-missing relative clause. OP↓ in (c) will be substituted with (b), creating a clause [*banana-lul coaha*] (

).

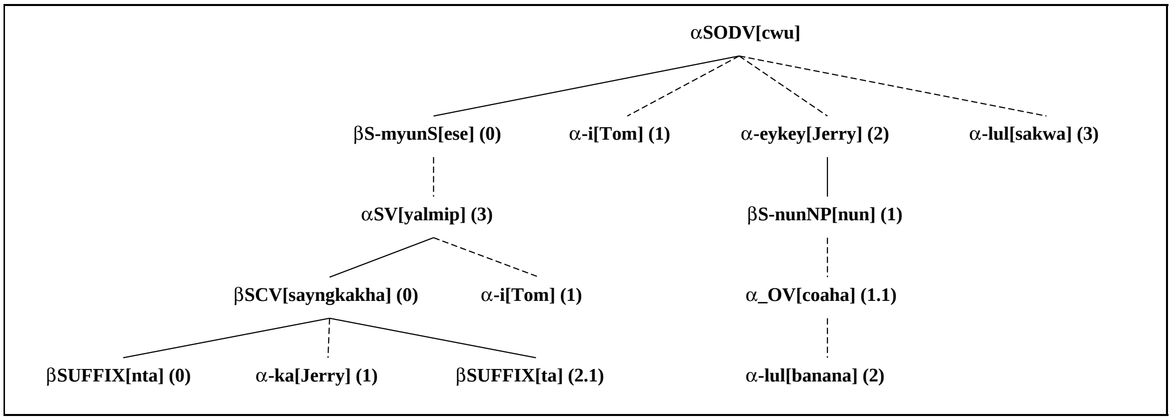
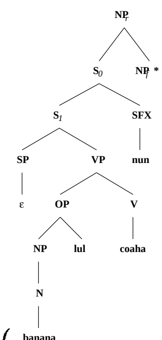
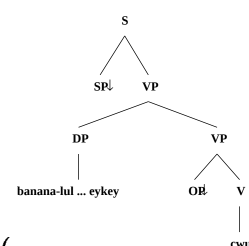


Figure 3.11: Derivation Tree

This partially derived tree again will be substituted into the node S_1 in (d), thus creating [*[banana-lul coaha-nun*], or [*who loves bananas*] (

).

This partially derived tree adjoins onto (e). Then, the tree, *banana-lul coaha-nun Jerry-eykey* with DP as a root node will be substituted into DP↓ node in (g) (

). SP↓ in (g) will be substituted with (a) or *Tom-i*, and OP↓ will be substituted with (f), *sakwa-lul*. So far, [*Tom-i banana-lul coaha-nun Jerry-eykey sakwa-lul cwu*][14] or [*Tom give*$_{stem}$ *an apple to Jerry who loves a banana*] is created.

Now, Figure 3.9(h) adjoins onto this partial derived tree; (h) introduces a new S_2 node, thus making it possible for the main clause to be attached to it. In other words, by adjoining (h), we have a sentence [*Tom-i banana-lul coaha-nun Jerry-eykey sakwa-lul cwu-ese*] or [*As*

[14] Still, it is in the state before the appropriate morphological form is adjoined to make a clause. Also, the order of parsing can be different, even though the final derived tree should be same.

Tom gave an apple to Jerry who loves a banana]. *Ese* in Korean functions both as a past tense of the verb *cwu* or *give* and as a conjunctive suffix. The last part of the sentence follows a similar pattern. Figure 3.11 is a derivation tree for sentence (17). Figure 3.10 is the final derived tree for sentence (17).

3.12 Application to the Military Messages

The Korean grammars presented here were applied to the military telecommunication messages. This data is supplied by Mr. Yaeger, US army research officer, and translated by Sungki Suh and Jeyhoon Lee, at the University of Maryland. Some of the input data is listed below[15], and the rest of the data is shown in Appendix A.

⌈ COMMO IS UP. PLEASE SEND A CURRENT CMDRS REPORT ON ALL UNITS.
통신이 이제 가능하다. 모든 부대에 현재 지휘관 보고를 보내라.

THE ONLY UNIT WE HAVE A CMDR REPORT ON IS THE 149, WE ARE NOT IN CONTACT VIA MCS WITH THE 67, 69 OR 1-167.
지휘관 보고가 입수된 유일한 부대는 149이다. 67, 69, 그리고 1-167 과는 기동제어시스템을 통한 접촉이 되고 있지 않다.

REQUEST THE FOLLOWING ITEMS BE RESUPPLIED IN THE FOLLOWING QUANTITIES:
다음의 항목들이 지시된 양으로 재공급될 것을 요망한다 :

DIESEL 25,000 GAL, FOOD 20 TONS, WATER 15,000 GAL, 155 DPICM. V 500RDS, VULCAN 10,000RDS, DECOM SUPPLY 50 TONS. ⌋

Previous messages were tested and parsed using XTAG 1.0. Around 100 elementary trees were needed for parsing the sentences in Appendix A.

In doing so, I used a Yale Romanization Code, and broke down the suffixes from the verb stem, and also separated particles from the noun, i.e., there is a blank between a noun and its particle in the input, as the actual Korean characters cannot be yet read from XTAG 1.0 system. The actual input data that was used is given below.

⌈ PLEASE SEND A CURRENT CMDRS REPORT ON ALL UNITS.
motun pwuntay uy choykun salyengkwan pokose lul ponay la .

WE ARE NOT IN CONTACT VIA MCS WITH THE 67, 69 OR 1-167.
67 , 69 , 1-167 kwa nun kitongceye sisteym ul tongha n cepchok i toy ko iss ci anta.

REQUEST THE FOLLOWING ITEMS BE RESUPPLIED IN THE FOLLOWING QUANTITIES: DIESEL 25,000 GAL, FOOD 20 TONS, WATER 15,000 GAL, 155 DPICM. V 500RDS, VULCAN 10,000RDS, DECOM SUPPLY 50 TONS.
taum uy hangmok tul i cisitoy n yang ulo caykongkuptoy l kess ul yomangha nta. ⌋

[15] Some of the translation might not be appropriate for the actual military messages.

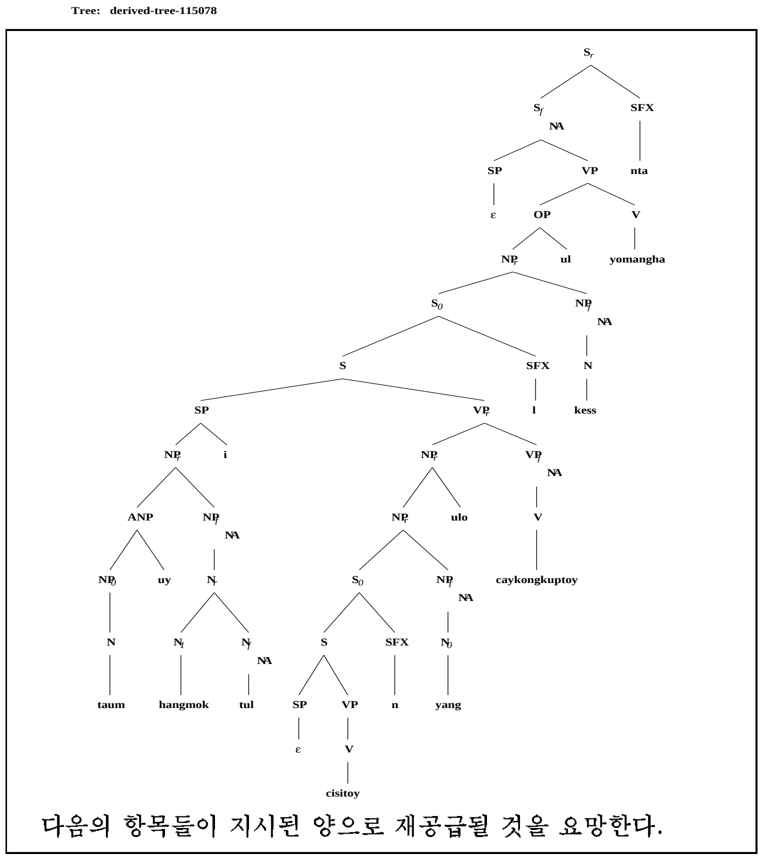
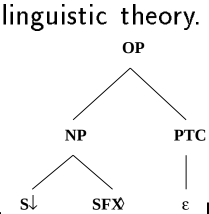


Figure 3.12: Army Data Derived Tree

Most of the sentences in Appendix A were parsed[16] as all the military telecommunication had repetitive and relatively simple patterns. If there was a problem, it was not due to the formalism of TAGs. Rather, it was a problem of controversial Korean linguistic theory. For example, I followed Choi's theory to represent ci (지) in the TAG tree as 

[10]. As Choi's theory itself is controversial, some people would assume a different representation for the lexical item, ci.

Most of the problems arose from handling scrambling, conjunctive structure, and case overlapping. Also, as the domain was telecommunication messages, the dropped arguments both in Korean and English were prevalent. In English, only the subject is dropped, whereas in

[16] In Appendix A, footnotes, explaining why it could not parse, were attached for each sentence which could not be parsed.

Korean, the problem was more serious as virtually any kind of argument can be dropped as long as it can be recoverable from the context. XTAG 1.0 is not yet augmented with a discourse model so that all the dropped arguments were lexicalized with ϵ. Some of the derived parse trees for the military messages are shown in Appendix B.

3.13 Summary

The Korean TAG grammars were applied to the military telecommunication messages. The derived trees of US Military messages are shown in Appendix A.

In the application to Telecommunication Messages, some of the representation of the TAGs looked awkward to some people. For example, I used an adjunction type tree for the conjugational suffix, and it was adjoined to the sentence structure. One can argue that the morphological form should be adjoined onto the verb phrase node (VP), instead of the S node[17]. However, I felt that the conjugational form of the verb might not just change the meaning of the verb, but it actually changes the meaning of the whole clause. This concept can be controversial.

There are some other places that are arguable. However, the LTAG grammar for Korean given here is preliminary and should be viewed as such; it meets the base requirements of LTAG, namely, encapsulation of predicate argument structures and factoring recursion from the domain of dependencies. In my work here I do not compare my grammar with corresponding grammars in other formalisms[11]. Some of the trees may look arbitrary and indeed may be so as they were motivated by the particular texts in the domain. Further study will help remove this arbitrariness.

[17] There are two positions on how to view verbal suffixes in the standard syntactic theories. One is to see a suffix as a functional category where all kinds of verbal suffixes are viewed to head their own projection. The other position is a view that these are incorporated into the head of V.

Chapter 4

Korean-English Machine Translation

Machine Translation was conceived in the 1950's, and there was considerable research afterwards. However, it waned in the '70s as the complex nature of linguistic structure brought discouragement. Now, interest is beginning to revive.

It is well-understood that accurate machine translation often requires reference to contextual knowledge for the correct treatment of linguistic phenomena such as pronoun reference and gender agreement[15]. This is still, in many cases, an unsolved problem for natural language analysis[39], which adds to the burden of the already beleaguered machine translation systems[14].

In this section I present a prototype system for machine translation between English and Korean which is implemented in Synchronous TAGs[51]. Essentially, it makes use of the grammars for Korean described in the previous chapter. Although this is essentially a transfer based approach, it uses feature unification for lexical selection and is being augmented with a discourse model to handle discourse related phenomena such as recovery of topicalized arguments.

[0]† This chapter has been published as "Korean To English Translation Using Synchronous TAGs" in the Proceeding of the First Conference of the Association for Machine Translation in the Americas, with minor modification.

4.1 Synchronous TAGs

Synchronous Tree Adjoining Grammars (STAGs) are a variant of TAGs introduced by Shieber and Yves Schabes to characterize correspondences between tree adjoining languages[51]. They can be used to relate TAGs for two different languages for machine translation[2], for relating a syntactic TAG to a semantic one in the same language[51, 1], for generation[52], or for semantic analysis. STAGs have been shown[2, 24] to be capable of handling syntactic and lexical-semantic divergences shown by Dorr[13][1].

Transfer rules are correspondences between nodes of the elementary trees of a TAG associated with lexical entries, and this allows lexical transfer rules to be defined over a large domain of locality. The transfer lexicon puts into correspondence a tree from the source grammar instantiated by lexical insertion with a tree from a target grammar. The source sentence is first parsed according to the grammar for the source language. Each elementary tree in the source derivation tree is then mapped to a tree in the target derivation tree by looking in the transfer lexicon. These trees are combined according to the links specified between the nodes in the correspondence trees, and the target sentence is read off the final target derivation tree. Correspondences can be made between trees, lexical items, or individual features.

Using STAG, the transfer between Korean and English can be done directly by putting into large elementary correspondence units without going through some interlingual representation and without major changes to the source and target grammars. Lexical transfer rules can be defined to avoid the defects of a mere word-to-word approach but still benefit from the simplicity and elegance of a lexical approach[2].

4.2 An Example

The translation process consists of three steps in which the generation step is reduced to a trivial step. The source sentence is parsed accordingly to the source grammar. Each elementary tree in the derivation is considered with the features given from the derivation through unification. Second, the source derivation tree is transferred to a target derivation. This step maps each

[1] Dorr shows five kinds of syntactic divergences that can be accounted for by means of parameterization: constituent order, preposition standing, long-distance movement, null subject, and dative. TAG's extended domain of locality allows it to handle these types of divergencies relatively easily.

elementary tree in the source derivation tree to a tree in the target derivation tree by looking in the transfer lexicon. And finally, the target sentence is generated from the target derivation tree obtained in the previous step[2]. As an example, consider the fragments of the transfer lexicon given in Figure 4.1. The canonical sentences to translate between two languages are generally transitive sentences. Korean is an SOV language while English is SVO, so there are some structural differences between the two languages[2]. Figure 4.1 shows the links between the transitive trees for the sentence (18)

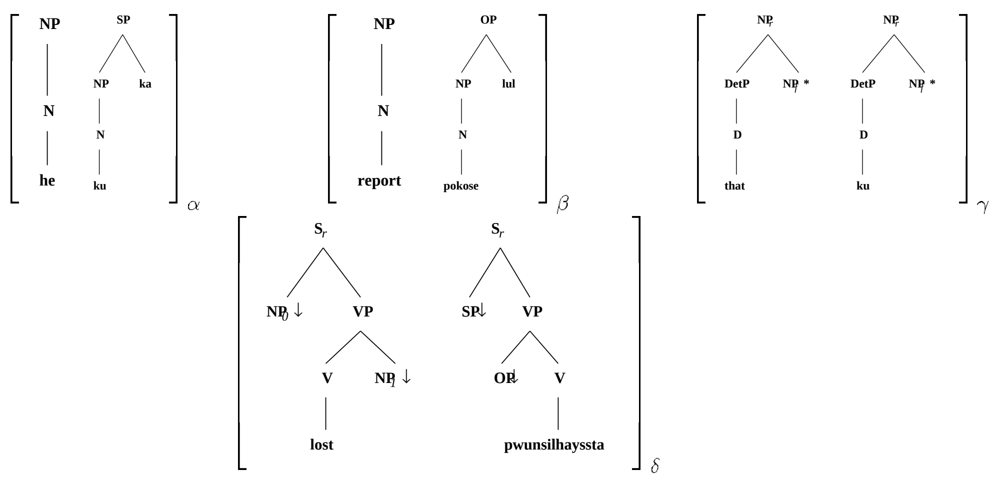


Figure 4.1: The Korean-English Transfer Lexicons

18	ku-ka	ku pokose-lul	pwunsilha-yssta.
	he-NOM	that report-ACC	lose-PAST
	그가	그 보고서를	잃어버렸다

The transfer lexicon consists of pairs of trees, one from the source language and one from the target language. Within the pair of trees, nodes may be linked. Whenever in a source tree, adjunction or substitution is performed on a linked node, the corresponding tree paired with it operates on the linked node. For example, we start with the pair δ and we substitute the pair α on the link from the Korean node SP to the English node NP_0. This operation yields the derived pair δ_1. Then, if pair β is substituted into the NP_1-OP pairs in δ_1,δ_2 are

[2]This is an instance of what Dorr would call constituent order divergence[13].

generated. Again, from δ_2, by adjoining onto pair γ in the NP-NP pairs, δ_3 is created, thus correctly transferring sentence (18)[3].

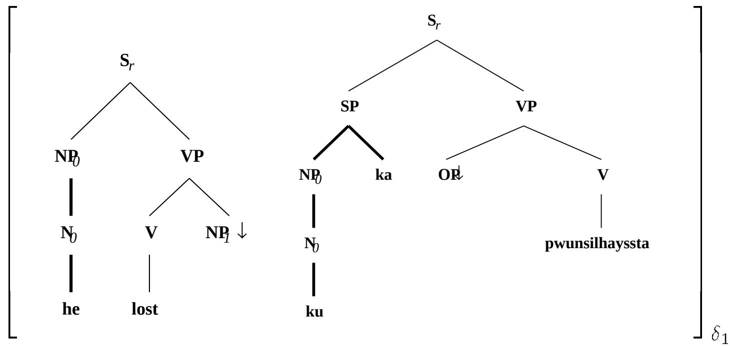


Figure 4.2: Derived δ_1 tree

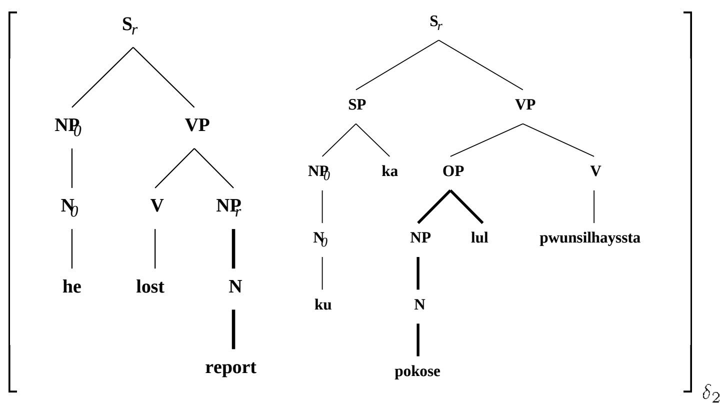


Figure 4.3: Derived δ_2 tree

In order to transfer both the predicate-argument relations, and the construction types, it is necessary to be able to refer to a specific tree in a tree family. This is done by matching the syntactic features by which the different trees are identified within a tree family. When a syntactic feature of a given tree family does not exist for the corresponding tree family in the target language, it will be ignored.

It is not a problem when an elementary tree of a certain constituent structure translates into an elementary tree with a different constituent in the target language. Furthermore, elementary

[3] For the simplicity of the argument, the suffix node SFX was not specified here

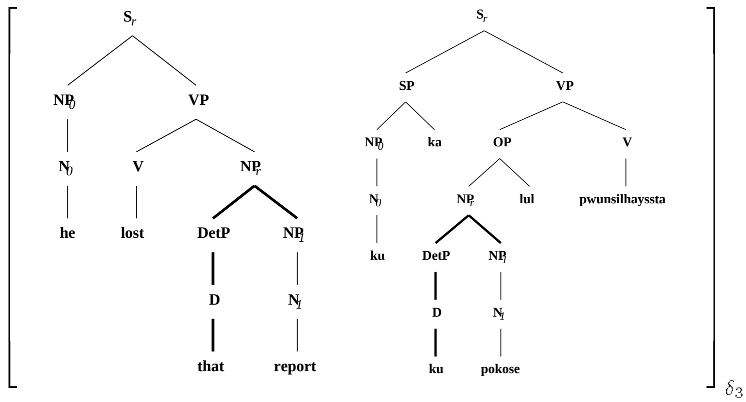


Figure 4.4: Derived δ_3 tree

structures of the source language need not exist in the target language as elementary structures. Some Korean predicates do not have the same number of arguments as their corresponding English ones. In such cases, the pair does not consists of pairs of elementary trees but rather, as pairs of derived trees of bounded size. Since the match is performed between derived trees, no new elementary trees are introduced in the grammars[4].

4.3 Relative Clauses

Relative clauses in Korean are relatively straight-forward using STAGs. Sentence (19) shows an example sentence in Korean that uses a relative clause. The verb *ssun* (*write*) is in its adnominal form, which indicates that the embedded S is a relative clause. Note that the word being modified, *pokose* (*report*), is on the right side of the relative clause, whereas in English, the relative clause is on the left.

	ku-ka	[kunye-ka	ssun	ku pokose-lul]	pwunsilhayssta.
19	he-NOM	[she-NOM	write-ing	that report-ACC]	lose-PAST
	He	lost	that report	she	wrote.

Since the adnominal form indicates that the embedded S is a relative clause, it will select the Korean relative clause tree shown in Figure 4.5 when parsed. The relative clause tree for

[4] For example, in Korean, there is no article, and every English noun phrase with article should be translated into Korean noun phrase without article. This means the derived tree of English noun phrase with article will be paired with the derived tree of Korean Subject or Object phrase elementary substituted with particles.

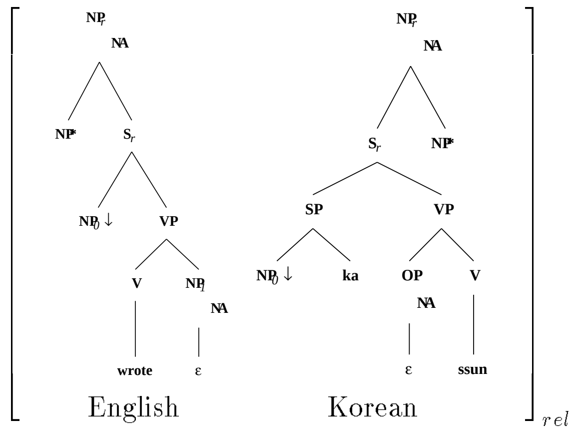


Figure 4.5: Relative Clause

English is also shown in Figure 4.5, and the STAG linkings are given between the two trees. From here the translation is entirely straight-forward, using the linkings between the trees.

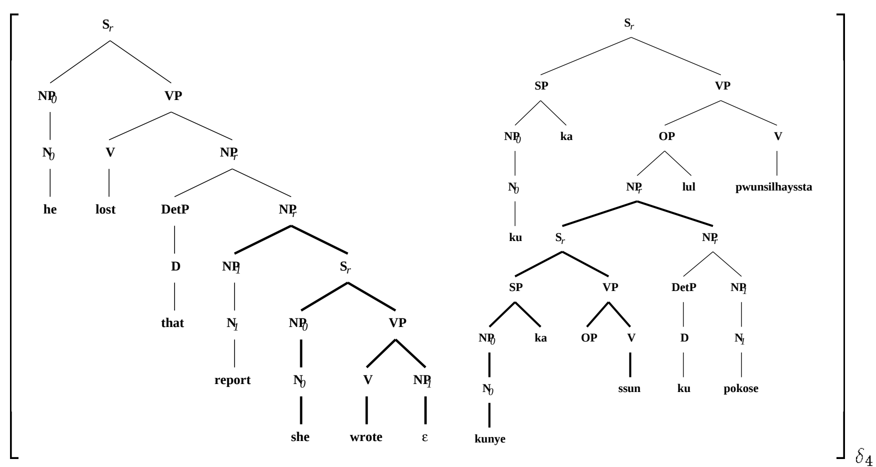


Figure 4.6: Derived δ_4 tree

Onto the NP node in the δ_3 pair, the pair in Figure 4.5 will be adjoined for each language. Figure 4.6 is the final derived tree pair for translating Korean sentence *ku-ka [kunye-ka ssun ku pokose]-lul pwunsilhayssta* into the English sentence *He lost that report [she wrote].*

4.4 Matching Feature Structures

How features can be transferred or interpreted plays a vital role in Synchronous TAGs. However, some of the features in one language has no equivalent features in the other. For example, agreement between subject and verb is not necessary for Korean, whereas in English, agreement[5] assigns a nominative case. So, *kata* (go), a present form of the verb, in Korean has only one morphological form regardless of the person of the subject whereas in English, *goes*, or *go* can be used,i.e., it depends upon whether it is First Person, Second Person, or Third Person. So, as is shown in Figure 4.7, the *case* feature need not to be considered in Korean whereas *tense* feature will be considered both in English and Korean, and will be linked together.

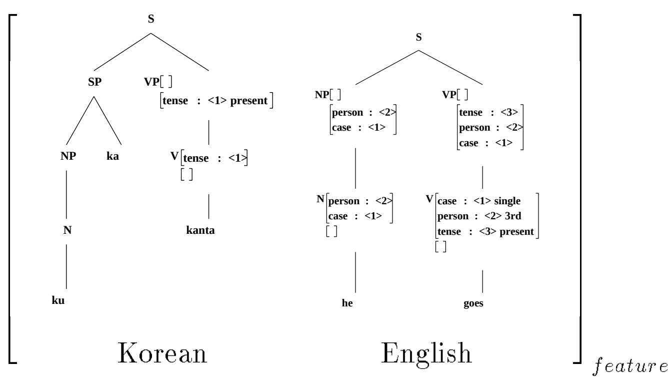


Figure 4.7: Feature Transfer

Another place where feature plays a vital role is in semantic interpretation and attribute phrases transfer of English. Korean requires attribute phrases to be realized as full relative clauses, not as simple modifying expressions. This can be easily seen in the structure of PP modifying NP as can be seen in sentence (20). The translation of this kind can be achieved by mapping one English PP structure to several Korean structures depending upon the semantic features of the NP phrase.

[5] as in government and binding theory[11]

	E:	Tom	saw	the apple	on	the table.
	K1:	Tom-i	takca-wiey	iss-nun	sakwa-ul	poassta.
		Tom-Nom	table-Loc	exist-REL	apple-Acc	see-PAST
20		톰이	탁자위에	있는	사과를	보았다
	K2:	*Tom-i	takca-wiey		sakwa-ul	poassta.
		*Tom -Nom	table-Loc		apple-Acc	see-PAST
		*톰이	탁자위에		사과를	보았다

4.5 Wh Questions

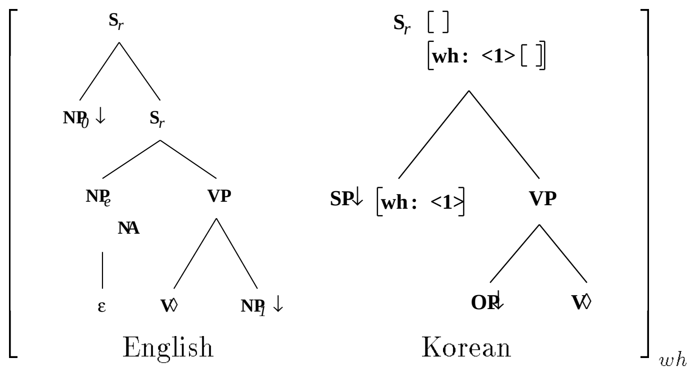


Figure 4.8: Mappings for [wh+] subject trees

Translating wh-questions from Korean to English is slightly more interesting than from English to Korean because wh-items do not move in Korean as they do in English. Hence, while the basic indicative tree suffices for the wh-question in Korean, a separate structure is required for English. When translating from English to Korean, the English wh-tree simply maps to the Korean indicative tree. However, when translating from Korean to English, the Korean indicative tree must map to one of several possible trees in English. This depends on whether the sentence has a wh-word in one of its arguments, and if so, which argument.

This highlights one of the strengths of STAGs, since we can take advantage of the feature structures in specifying the mapping. When the mapping of the basic Korean indicative tree to the non-wh English tree is specified, all of the NP nodes in the Korean tree are required to have the feature [wh-]. There are separate entries for the Korean trees with the a [wh+]

subject, and a [wh+] object, which respectively map to the wh-subject-extraction tree and wh-object-extraction trees for English. Figure 4.8 shows the mapping between Korean and English for [wh+] subjects.

4.6 Passive Structure

Korean has four different stem-forming suffixes : *-i, -hi, -li, -ki* for representing passives. We need to extend the domain of passives enough to cover *-tang/-pat-* constructions and the suffix *-ci*. The -tang/-pat construction occurs only when the lexical category of verb and noun are associated with each other. According to Chang-In Lee, we need to choose *-tanghata* instead of *-twoeta* for machine translation[31]. The choice is made because *-tanghata* can be put anywhere if *-twoeta* can be used, but not vice versa. Here are some of the possible types of Korean passives.

Jerry-ka	mwuncey-lul	pwulta.	→	Mwuncey-ka	pwul-li-ta.
Jerry-NOM	problem-ACC	solve.		problem-Nom	solve-PASS-DECL
제리가	문제를	풀다		문제가	풀리다
Tom-i	Jerry-lul	cap-ta.	→	Jerry-ka	cap-hi-ta.
Tom-Nom	Jerry-ACC	catch-DECL.		Jerry-Nom	catch-PASS-DECL.
톰이	제리를	잡다		제리가	잡히다
Tom-i	Jerry-lul	salangha-ta.	→	Jerry-ka	salang-patta.
Tom-Nom	Jerry-ACC	love-DECL.		Jerry-Nom	love-PASDECL.
톰이	제리를	사랑하다		제리가	사랑받다
Jerry-ka	Tom-ul	cheyphoha-ta.	→	Tom-i	cheypho-tang-hata.
Jerry-Nom	Tom-ACC	arrest-DECL		Tom-Nom	arrest-PASDECL
제리가	탐을	체포하다		톰이	체포당하다
Tom-i	yuli-lul	kkayta.	→	Yuli-ka	kkay-ci-ta.
Tom-Nom	glass-ACC	break-DECL.		glass-NOM	break-PASDECL.
톰이	유리를	깨다		유리가	깨지다

-Tanghata/-patta can be termed lexical passives while *-Ki* suffix can be called a morphological passive. In other words, *-tanghata* and *-patta* can occur as independent verbs with the meaning of *to undergo* and *to receive*, respectively. *-Ci* represents the intensity of the resistence of the *Patient*. Korean is more specific, or rather more meaning bounded whereas in English, the bounds are more syntactic. For example, the passive of *Tom loves Jerry* would be *Jerry is loved by Tom*, but not *Jerry is loved from Tom*, whereas in Korean, as long as it can reflect an agent role, any kind of marker can be attached to Tom. So, in *Jerry-ka Tom-X*

salang-pata, X can be either *-ulo pwute* (from) or *-e uyhay* (by).

4.7 Lexical Selection

To achieve the goal of the MT system, we have to handle word meanings, including the task of "disambiguating" word senses, and unknown usages, both in the source and target languages. The task of lexical selection in machine translation consists of choosing the target lexical item which most closely carries the same meaning as the corresponding item in the source text. This cannot be solved by the straightforward, albeit tedious, process of simply listing corresponding verb pairs for the source and target languages, since there is not always a one-to-one correspondence, and accurate lexical selection can sometimes depend on very subtle semantic distinctions or even require reference to the context[61].

The problem of precisely capturing the semantic distinctions required for accurate lexical choice is not usually noticeable in small systems, but comes into play when expanding the lexical base to include more fine-grained senses of basic concepts. For example, the semantic features that are used in selecting the correct serial verb construction in Chinese (such as the initial shape of the object, choice of instrument) are not all used in selecting English verb senses. The end result is that lexical selection is often predicated on the existence of semantic features that are completely irrelevant to the source language.

We can find two different senses of *lose* that use the same lexical item in English TAG system, but require distinct Korean expressions, as seen in sentences (21) and (22).

	He	lost	that	report
21	ku-ka	ku	pokose-lul	pwunsilhayssta.
	he-NOM	that	report-ACC	lose-PAST

	He	lost	that	battle.
22	ku-ka	ku	centwu-eyse	ciessta
	he-NOM	that	battle-LOC	lose-PAST

Not only does our system need to correctly select between these two senses, but it also must handle a structural divergence. This is straightforward in STAG, since the tree mapping maps the relevant NPs to each other as seen earlier. The lexical selection issue is more interesting. In Korean, the first sense of *lose,* 'to mislay', is *pwunsilhayssta,* which selects for physical objects. The second sense *ciessta*,'to fail to win,' takes an optional PP whose NP is a competitive noun. English *lose* does not make this syntactic distinction. Thus the mapping from Korean into English is quite simple, as the two verbs *pwunsilhayssta* and *ciessta* both simply map to transitive *lose*. Going from English to Korean is more difficult, and we will briefly switch the direction of translation to show how this can be handled.

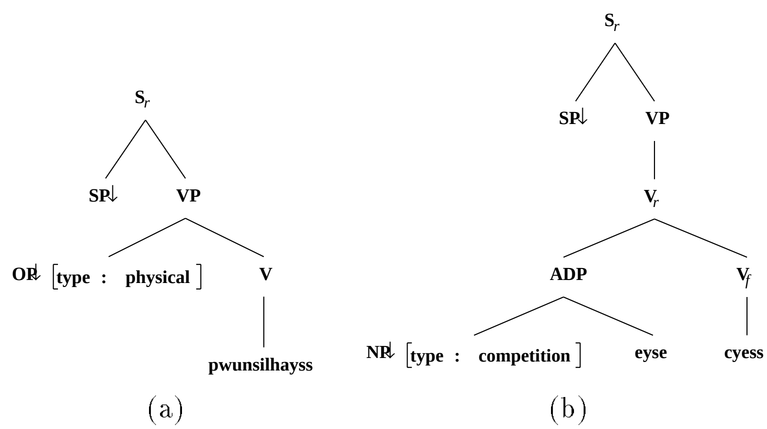


Figure 4.9: *pwunsilhayssta* and *ciessta* trees

Since in the XTAG implementation, the English *lose* makes no syntactic distinction between physical and competitive arguments, it is not necessary to include them in the English feature structure. Instead, we simply allow the English *lose* to translate ambiguously into both *pwunsilhayssta* and *ciessta*. The Korean verbs, which do make this distinction, will impose selection restrictions on the nouns associated with them. If we translate *He lost that report* into Korean, we will initially get two trees, as seen in Figures 4.9a and 4.9b, corresponding to the two verbs with their associated selection restrictions. *Pwunsilhayssta* requires its object to be [type: physical] and *ciessta* requires that the object of any PP that adjoins on must be [type: competition]. When *pokose* (*report*) substitutes into the NP position, it carries with it information on its semantics, specifically, that it is [type: physical]. When the trees have been built and the top and bottom features on each node try to unify, the semantic features on the *ciessta* tree will clash, and the tree will fail. The only translation for *He lost that report* is *ku-ka pokose-lul pwunsilhayssta*, as seen in the tree in Figure 4.9(a).

4.8 Using LCS in Synchronous TAGs

4.8.1 Divergences

A translation divergence arises when the translation of a source language into a target language results in a very different form. There are several divergence types: conflational, structural, thematic, categorial, demotional, promotional, and lexical[13]. Between Korean and English also, several divergence types can be easily seen.

- Structural Divergence :

K:	Tom-i	pang-ulo	tule-ka-ssta
	Tom-NOM	room-LOC	enter-go-PAST
	톰이	방으로	들어갔다
E:	John	entered	the room

  The verbal object is realized as a noun phrase in English whereas it is realized as a noun + Locative particle in Korean.

- Conflational divergence :

K:	Tom-i	Jerry-eykey	towum-ul	cwuessta.
	Tom-Nom	Jerry-Dative	help-ACC	give-PAST
E:	Tom	helped	Jerry.	

  Conflation is the incorporation of necessary components of meaning of a given action. Korean uses *-eykey towum-ul cwuessta* as *help* in English.

- Categorial Divergence :

K:	Tom-un	khi-ka	khuta
	Tom-TOP	height-NOM	tall-DEC.
E:	Tom	is	tall.

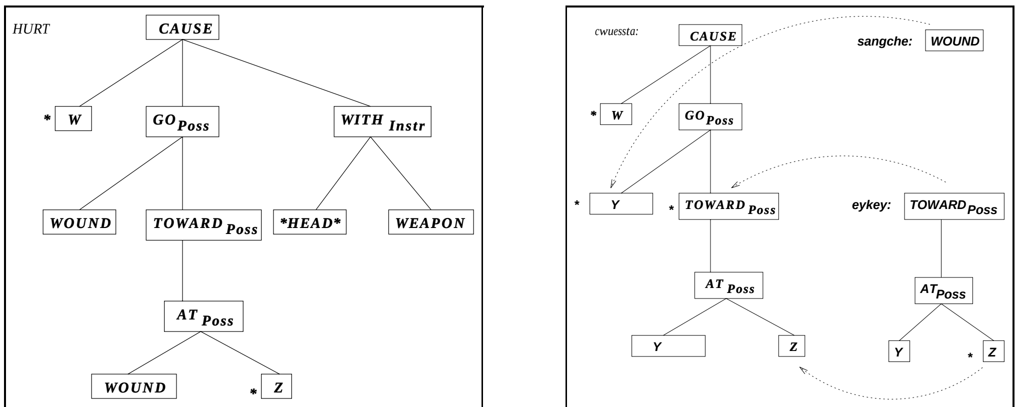
  The predicate is adjectival in English, whereas it is verbal in Korean.

Abeillé et al. argue that in LTAGs, the transfer between two languages can be done by putting directly into correspondence large elementary units without going through some interlingual representation and without major changes to the source and target grammars. This is due to LTAG's extended domain of locality and semantic dependencies. This allows the possibility that an explicit semantic representation level can be avoided[2].

On the other hand, Dorr proposes that Lexical Conceptual Structures offer a major advantage in their capability to provide an interlingua form that can handle divergences.

At this moment, I am not in a position to argue whether an interlingua form should be used or not in STAGs for machine translation. Even if an interlingua form is to be used, I cannot argue whether a certain interlingua such as LCS is better than another such as predicate logic. However, it is worth considering what should be done, or what the problem is, for using the LCS in STAGs as an interlingua.

4.8.2 An Example



(a)Tom hurt Jerry (b)Tom-i Jerry-eykey sangche-lul cwuessta.

Figure 4.10: Hurt Event Tree in English and Korean

	E:	Tom	hurt		Jerry.
23	K:	Tom-i	Jerry-eykey	sangche-lul	cwuessta
		Tom-NOM	Jerry-DAT	wound-ACC	give-PAST.

For sentence (23), English uses the single word *hurt* for the words *-eykey sangche-lul cwu-ta* (상처를 주다) in Korean. In Korean, *wound* or *Sangche*(상처) is overtly realized. The Experiencer is realized as Jerry-eykey(*to Jerry*) in Korean whereas in English, it was realized as an object (*Jerry*). The above sentence for both English and Korean can be represented by the same Lexical Conceptual Structure [6].

$[_{Event}$	CAUSE					
	$([_{Thing}$	*W],				
	$[_{Event}$	GO_{Poss}				
	$([_{Thing}$	WOUND],				
	$[_{Path}$	$TOWARD_{Poss}$				
	$([_{Position}$	AT_{Poss}	$([_{Thing}$	WOUND],	$[_{Thing}$	*Z])])])])]

[6]Tense and aspect are currently aspectual features on the composed LCS, usually derived from syntactic analysis (e.g., tense = present, aspect = progressive).

The meaning corresponds to "thing W causes thing Z to possess a WOUND." The * notation is used to specify the language-specific correspondence between LCS arguments and the syntactic structure, which is the second level of RLCS description[7][13]. The * notation represents a similar function of ◇ in TAGs, as ◇ is the place where the actual lexicon is realized.

In Figure 4.10, it can be noticed that in English, *hurt* can be represented with one structure whereas in Korean, including the structure of the verb *cwuessta*, two other structures should unify with each other without conflict.

4.8.3 Using the LCS in TAGs

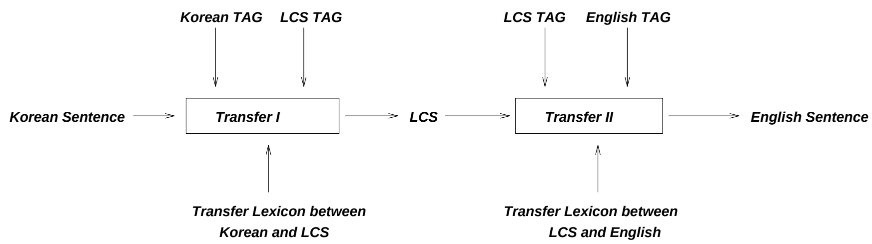


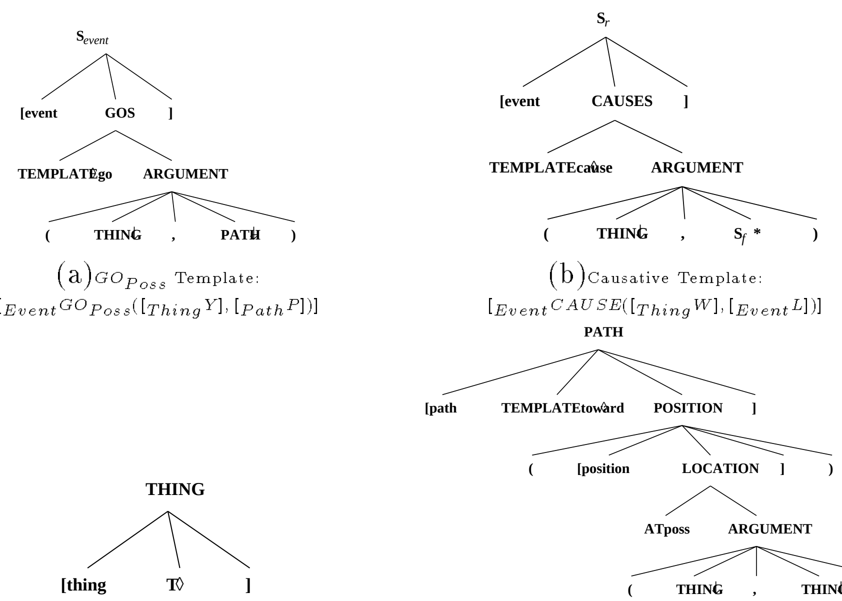
Figure 4.11: MT via LCS using STAGs

Figure 4.12 shows how each RLCS unit can be represented in TAGs. A scheme using LCS is shown in Figure 4.11. Using LCS as an interlingual form, a transfer lexicon between Korean and LCS is used for translating Korean to LCS and LCS to Korean. A transfer lexicon between LCS and English can be used again to translate LCS into English and vice versa. So, if we want to translate English to Korean, we use STAGs to translate English to LCS; then, using the LCS that was produced from the previous translation, STAGs is used again, to translate the LCS to Korean.

If each elementary RLCS type TAG tree shown in Figure 4.12 can be roughly mapped directly to both English and Korean elementary trees, it would be ideal and effective. As a matter of fact, LCS is easy to represent in TAGs as it has a smaller vocabulary and there are not many structures compared to natural languages such as Korean and English. However, there is a difference between the characteristics of a natural language and that of LCS.

[7] RLCS means Root LCS and CLCS means Composed LCS. See [13] for further discussion of RLCS and CLCS.

[7] ‡In representing TAG trees for RLCS, functional marks (e.g. (or ,) can be redundant for representing an LCS structure. However, to accept the LCS type string in TAGs, all the functional marks should be lexicalized. If we are not considering this, the representation in Figure 4.12 can be simpler.



(a) GO_{Poss} Template: $[_{Event} GO_{Poss}([_{Thing} Y], [_{Path} P])]$

(b) Causative Template: $[_{Event} CAUSE([_{Thing} W], [_{Event} L])]$

(c) Causative Template: $[_{Event} CAUSE([_{Thing} W], [_{Event} L])]$

(d) $[_{Path} TOWARD_{Pos}([_{Position} AT_{Poss}([_{Thing} Y], [_{Thing} Z])])]$

Figure 4.12: TAG trees for RLCS

Whereas it is generally easy to incorporate the concept of domain of locality into each elementary tree for natural language, it is not trivial to incorporate the concept of domain of locality into LCS elementary trees.

For example, in the case of Figure 4.10(b), it shows that the variable Y in the *cwuessta* structure and the variable Y in the *eykey* structure should be unified, even though they are from a different structure. Only in that way, when Y is lexicalized with WOUND, the variable Y in the *eykey* structure would not conflict with Y in the *cwuessta* structure.

This implies two conditions. One is that in TAGs notation, the elementary trees in Figure 4.12(c) and 4.12(d) cannot be mapped to Korean or English trees separately, as the domain of locality cannot be preserved. Secondly, a node in an elementary tree in one language should be sometimes linked to two nodes in another language.

One approach for satisfying the two conditions above would be to change the STAGs formalism itself so that two nodes in separate elementary trees can be globally coreferenced; and if one node is lexicalized, then the other globally coreferenced node should be lexicalized with the same lexical item. That kind of mechanism can be implemented by modifying the parsing algorithm to check if any two nodes are globally coreferenced. By this way, both

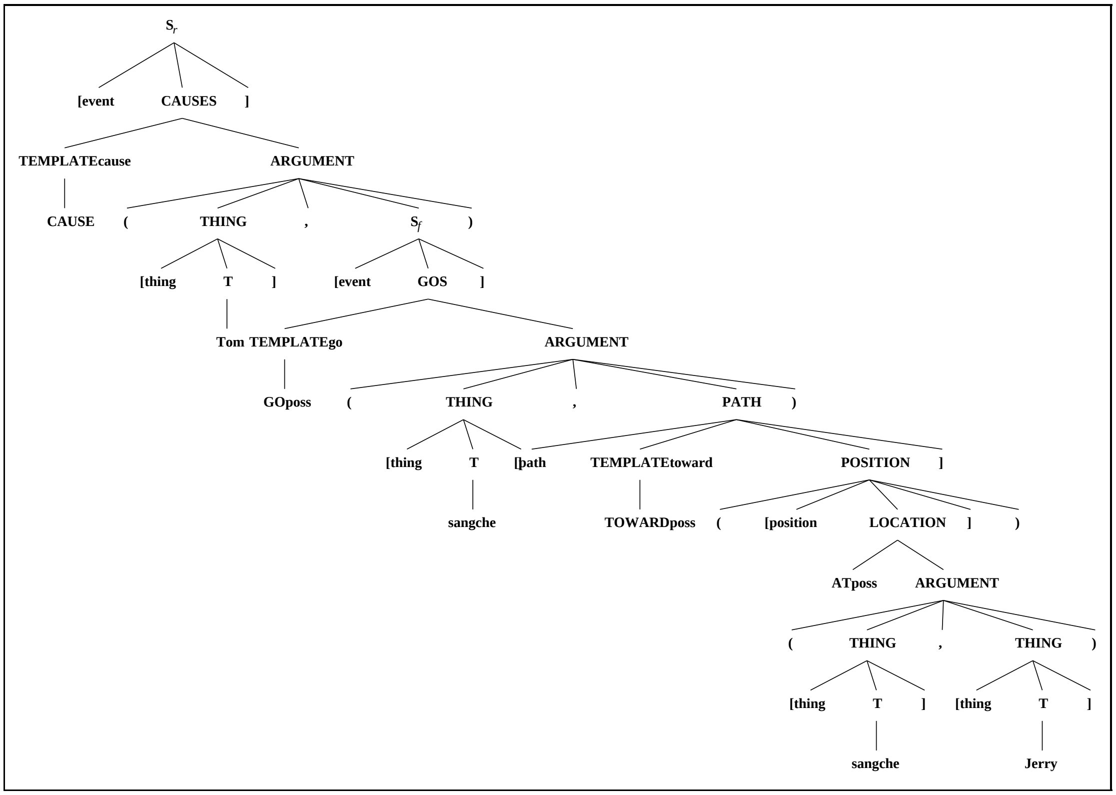


Figure 4.13: Hurt Event LCS form in TAGs

the two conditions can be satisfied. However, this involves a significant change to the TAG formalism and would require a thorough study of the extended power it is imposing, as well as its formalization.

The other option without changing the current TAGs formalism is to combine several elementary RLCS type TAG trees together up to the point where the domain of locality can be preserved, and link this whole structure to the Korean or English TAG tree.

This scheme is shown in Figure 4.14. In the case of mapping the English verb *hurt* to the LCS, *WOUND* itself is already lexicalized on the right tree. However, in the case of mapping the *cwuessta* verb structure in Korean to the LCS, the NP node is linked to either the THING_2 or THING_4 node in the right tree of the Korean-LCS tree, and they are coreferenced. As can be seen on the right tree of the Korean-LCS tree in Figure 4.14, the domain of locality can be preserved in this way, as the two nodes are represented in a single tree, and both trees (THING_2 and THING_4) should be colexicalized.

One definite advantage of using the LCS form in STAGs is when implementing multilingual

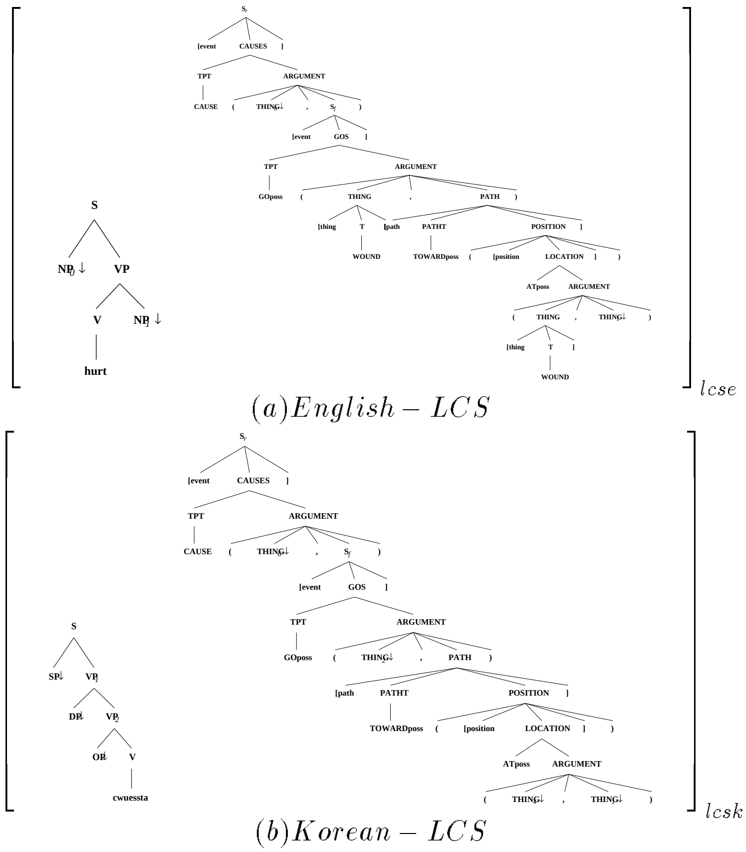


Figure 4.14: English-LCS and Korean-LCS Transfer Lexicons

translation mechanism for several languages. In other words, for all the languages that have been linked to LCS, the machine translation between the two languages among them will be trivial. For example, once a transfer lexicon between LCS and English is established, we do not need to worry about building transfer lexicons between English to any other natural language. Also, the translation between two languages will not be done by arbitrary interpretation of one person, i.e., one person can build a transfer lexicon between English and LCS, not knowing Korean at all, and the other person can build a transfer lexicon between Korean and LCS, not knowing English, at all. If we use a direct translation from English to Korean, the implementor should know both languages. Even so, the transfer lexicon can be arbitrary. Probably, the implementor will try the most convenient way of linking two languages in STAGs, which is not a real sense machine translation.

One of the historical arguments in favor of the interlingua approach has been that, since it revolves around a deep semantic representation, it is better able to handle pronoun reference and other linguistic phenomena that are seen as requiring a knowledge-based approach, even though recent implementations of machine translation systems are blurring the distinction between transfer systems and interlingua systems[8]. However, the claims about the advantages of an explicit semantic representation level still need to be further investigated.

4.9 Summary

To illustrate the coverage of the system and the effectiveness of semantic feature unification, I gave examples of various syntactic phenomena, e.g., relative clauses and wh-questions and semantic phenomena, e.g., accurate lexical selection for polysemous verbs. The alternative approach using an interlingua such as LCS in TAGs is also discussed. Implementing an interlingua form in TAGs can be done. However, more thorough research is needed for a more efficient method of the implementation.

[8]Wilks discusses the hybrid IBM system which achieves improved coverage by combining certain transfer based techniques with what was fundamentally a statistical approach[60]. Nirenburg is currently proposing a multi-engine system which will apply more than one approach to a sentence and then choose the best result[38].

Chapter 5

Scrambling

In Korean, arguments of a verb can occur in any order. Furthermore, arguments can occur outside of their clause, which is called long-distance scrambling. Since Saito[45] proposed that scrambling is an instance of $\bar{A}$-movement [1], there have been some debates concerning the handling of scrambling, not to mention the nature of scrambling itself. Later, he claimed that unlike English topicalization, scrambling as $\bar{A}$-movement does not create a semantically significant operator-variable relation[46]. The nature of scrambling and its definition is still controversial. However, the following quote from Lee's dissertation describes some of the characteristics of Korean scrambling.

"... I use the term scrambling both in its descriptive and technical senses: Descriptively, I define scrambling to be the possibility that arguments of verbs may be arranged in any order, i.e., free word order. Technically, scrambling refers to an operation which either derives non base word orders, or all the possible word orders including the base word order, depending on the particular analysis one adopts. I use the technical term scrambling to refer to an operation deriving non-base word orders... "[34]

In this chapter, I describe several ways of handling scrambling using different formalisms such as HGs[32] and CCGs[54], together with a computational method of handling scrambling using Multi-Component Tree Adjoining Grammars (MC-TAGs)[59].

[1] Scrambling as an adjunction operation either to IP or to VP at S-structure

5.1 Characteristics of Korean Arguments

One of the characteristics of Korean arguments is their ability to *scramble*, or move within the sentence. This scrambling is allowed as long as the verbs can still be correctly associated with their arguments. The only restriction is that it should be verb-final.

5.1.1 Local Scrambling

Consider the verb *sayngkakhanta*, or *think*. We would predict that scrambling could occur between the clausal argument (S_0) and the subject NP (NP_0, as well as locally within the embedded clause. The canonical form of a sentence (톰이 제리가 사과를 먹었다고 생각한다), or *Tom thinks that Jerry ate an apple*, is given in Sentence (24)[2], and its derivation tree in TAGs is given in Figure 5.1. The verb *sayngkakha-nta* has two arguments, a subject (SP) followed by an embedded clause (C). Also, the verb *mekess-ta* or ate has two arguments, a subject (SP) followed by an object (OP). The embedded clause is shown in brackets.

	Tom-i	[Jerry-ka	sakwa-lul	mekessta-ko]	sayngkakhanta.
24	Tom-NOM	[Jerry-NOM	apple-ACC	eat-PAST-COMP]	think-PRES-DEC
	톰이	제리가	사과를	먹었다고	생각한다

We would predict that scrambling could occur between the clausal argument (C_0) and the subject (SP_1), as well as localy within the embedded clause, i.e., between the subject (SP) and the object (OP_1). Sentence (25) shows the same sentence with the subject NP and the embedded clause scrambled, while Sentence (26) shows the elements within the embedded clause scrambled, as well.

	(Jerry-ka	sakwa-lul	mekessta-ko)	Tom-i	sayngkakhanta.
25	(Jerry-NOM	apple-ACC	eat-PAST-COMP)	Tom-NOM	think-PRES-DEC
	제리가	사과를	먹었다고	톰이	생각한다

	(sakwa-lul	Jerry-ka	mekessta-ko)	Tom-i	sayngkakhanta.
26	(apple-ACC	Jerry-NOM	eat-PAST-COMP)	Tom-NOM	think-PRES-DEC
	사과를	제리가	먹었다고	톰이	생각한다

The sentences in (24), (25), and (26) can be easily represented by introducing new elementary trees such as Figure 5.2(a') and (b'). Figure 5.2(a') represents a tree where the

[2] Korean has two subject markers *-ka* and *-i*, which are distributed according to the phonology of the lexical item that they mark. There is no difference in meaning.

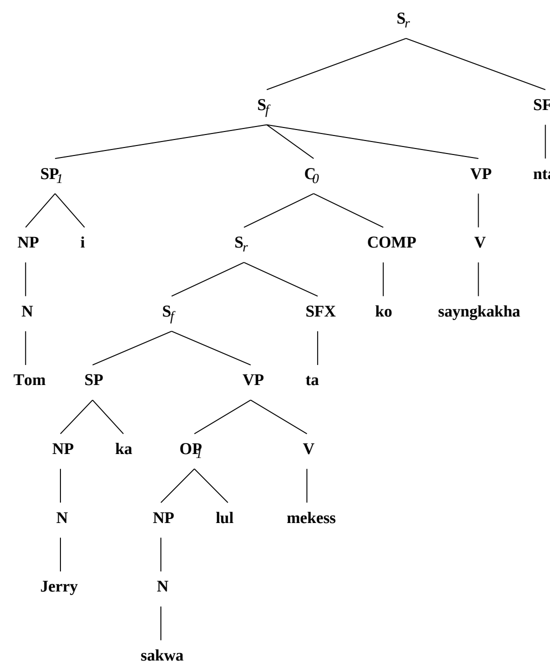
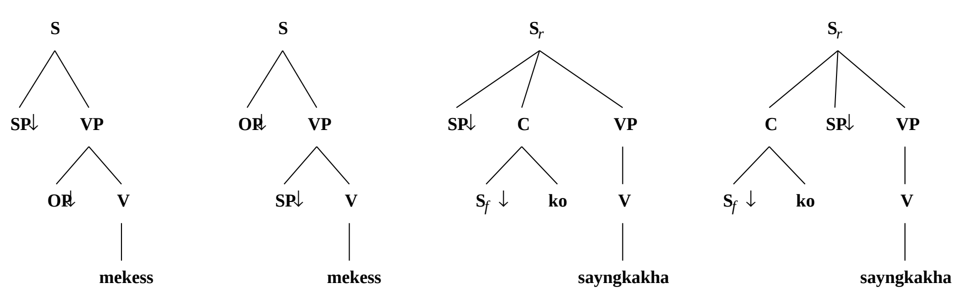


Figure 5.1: Tom-i Jerry-ka sakwa-lul mekessta-ko sayngkakhanta

position of subject and object are locally scrambled. Figure 5.2(b') represents a tree where the position of subject and the embedded clause are locally scrambled. Sentence (25) can be derived using the elementary trees in Figure 5.2(a) and (b'), while sentence (26) can be derived using the elementary trees in Figure 5.2(a') and (b').



(a) S + O + V (a') O + S + V (b) S + C + V (b') C + S + V

Figure 5.2: *sayngkakha* and *mekess* trees

5.1.2 Long-Distance Scrambling

Local scrambling could have been handled by providing all the possible elementary trees with the argument position scrambled for each verb. However, this approach has a major drawback, since Korean also allows certain permutations of arguments which amount to long-distance scrambling where elements can be scrambled outside of their clausal boundaries[34], even though this should not be taken to mean that all sentences in which long-distance scrambling has occurred will be judged equally acceptable.

Sentence (27) shows an example of this phenomena with *sakwa-lul*, or *apple*-ACC scrambled outside of its clause. Unfortunately, this cannot be represented using the trees in Figure (5.2.

27	sakwa-lul	Tom-i	Jerry-ka	mekessta-ko	sayngkakhanta.
	apple-ACC	Tom-NOM	[Jerry-NOM	eat-PAST-COMP]	think-PRES-DEC.
	사과를	톰이	제리가	먹었다고	생각한다
	Tom thinks	that Jerry	ate an apple.		

Furthermore, long distance scrambling is not affected just by the syntactic restriction. Although sentence (28) itself is well-formed, its meaning has changed; the subject arguments of the two verbs are reversed.

28	sakwa-lul	Jerry-ka	Tom-i	mekessta-ko	sayngkakhanta.
	apple-ACC	Jerry-NOM	Tom-NOM	eat-PAST-COMP	think-PRES-DEC
	사과를	제리가	톰이	먹었다고	생각한다
	Jerry thinks	that Tom	ate an apple.		

Scrambling, then, is not completely unconstrained. The verbs must still be able to correctly identify their arguments. That this is a semantic problem and not a structural one can be seen by comparing the sentences in (27) and (28) to the sentences in (29) and (30). The canonical form of the sentence is given in (29), while sentence (30) shows the long-distance scrambling version corresponding to sentence (28).

The structure is the same in the two examples, with the embedded subject scrambled to the beginning of the sentence, but in sentence (30) the meaning does not change. The semantic restrictions on the subject of the verb *sayngkakhanta* (*think*) prohibit it from taking *hankul-i* (*Korean*-NOM) as its subject, while the semantic restrictions on *elyepta-ko* (*difficult*-BE-COMP) prohibit it from taking a human (*Tom*) as its subject. The NPs then, can be scrambled in any order.[3]

	Tom-i	[hankul-i	elyepta-ko]	sayngkakha-nta.
29	Tom-NOM	[Korean-NOM	be-difficult-COMP]	think-DECL
	Tom thinks	that Korean	is difficult.	

	[hankul-i	Tom-i	[elyepta-ko]	sayngkakha-nta.
30	Korean-NOM	Tom-NOM	[be-difficult-COMP]	think-DECL
	Tom thinks	that Korean	is difficult.	

Clearly, scrambling is constrained by pragmatic, processing, and semantic factors. The contextual and semantic restrictions on word order do not translate into general rules that would categorially rule out certain formally definable orders, irrespective of the particular choice of lexemes and context.

5.2 Handling of Scrambling Using MC-TAGs

TAGs and related formalisms, due to the extended domain of locality of these formalisms, can combine a lexical head and all of its arguments in a single elementary structure of the grammar. However, Becker and Rambow show that TAGs that obey the co-occurrence constraint cannot handle the full range of scrambled sentences[4] [6][5].

The concept of Multi-Component TAGs was originally discussed by Weir[59]. There are three different definitions for MC-TAG (ignoring the issue of dominance links for the monment) depending upon the exact definition of adjunction: tree-local MC-TAG, set-local MC-TAG, and non-local MC-TAG. Weir defines non-local MC-TAG, in which trees from one set must be adjoined simultaneously anywhere into a derived tree[5].

[3] For the sentence *Tom-i Jerry-ka sakwa-lul mekessta-ko sayngkakhanta*, the following orders are the possible combinations.

a)	Tom-i	Jerry-ka	sakwa-lul	mekessta-ko	sayngkakhanta.
b)	Jerry-ka	sakwa-lul	mekessta-ko	Tom-i	sayngkakhanta.
c)	Tom-i	sakwa-lul	Jerry-ka	mekessta-ko	sayngkakhanta.
d)	Sakwa-lul	Jerry-ka	mekessta-ko	Tom-i	sayngkakhanta.
e)	Sakwa-lul	Tom-i	Jerry-ka	mekessta-ko	sayngkakhanta.
f)	*Sakwa-lul	mekessta-ko	Jerry-ka	Tom-i	sayngkakhanta.
g)	*Jerry-ka	sakwa-lul	mekessta-ko	sayngkakhanta	Tom-i.
h)	*Sakwa-lul	Jerry-ka	Tom-i	mekessta-ko	sayngkakhanta.
i)	*Tom-i	Jerry-ka	mekessta-ko	sakwa-lul	sayngkakhanta.

(f) is against verb-final condition, as *Jerry-ka* is behind its verb *mekessta*. (g) is also against verb-final condition, as *Tom-i* is behind its verb *sayngkakhanta*. (h) is impossible as the meaning has changed. (i) is against verb-final condition, as *sakwa-lul* is behind its verb *mekessta*.

[4] Tilman Becker and Michael Niv show that no formalism in the class LCFRS (which includes TAG) can derive scrambling[7][4].

[5] thanks to Tilman Becker for the comment

Later, non-local MC-TAG-DL (Multi-Component TAG with Dominance Links) was proposed as a way of handling scrambling in TAGs, where immediate dominance between nodes in elementary trees is relaxed[6]. In non-local MC-TAG, trees from a tree set are adjoined into the derived tree. There is no restriction on the locus of adjunction for each individual tree. An additional constraint system called *dominance links* was added, giving rise to MC-TAG-DL. A dominance link[6] may be specified between any two nodes of different trees in the same tree set. In the derived tree, the first node must dominate the other. According to Becker and Rambow, the definition of MC-TAG-DL is as follows[7].

"A MULTICOMPONENT TAG WITH DOMINANCE LINK (MC-TAG-DL) is a 5-tuple G = (V_N, V_T, S, I, A) where V_N and V_T are finite sets of nonterminals and terminals, respectively., S is a distinguished non terminal, I is a finite set of initial trees, and A is a finite set of finite sets of auxiliary trees. Within each set, dominance links between trees may be defined. Adjunction is defined to be the adjunction of an elementary auxiliary tree set into a derived tree [8][6]."

From now on, otherwise stated, MC-TAG refers to non-local MC-TAG-DL.

5.2.1 Method $\mathcal{A}$: A Linguistic Approach

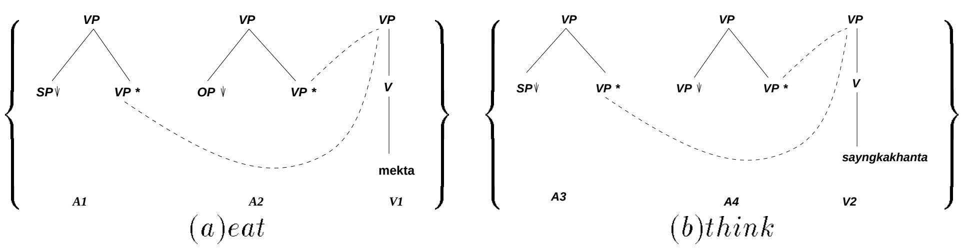


Figure 5.3: Set notation for coahanta and sayngkakahanta verb

[6]A dominance link between two trees β_1 and β_2 of the same tree set is a constraint on their multicomponent adjunction: after (simultaneous) adjunction of β_1 and β_2 into the same tree the foot node of β_1 dominates the root node of β_2.

[7]Later, Owen Rambow slightly changed the definition of MC-TAG-DL and renamed it V-TAG[43]. In V-TAG, there are no restrictions on adjunction sites. Trees from one tree set can be adjoined anywhere in the derived tree, and they need not be adjoined simultaneously or in a fixed order.

[8]Hockey and Srini have demonstrated the use of FTAGs in place of tree-local MC-TAGs[22]. Their approach sets up feature equations in elementary trees such that an adjunction of a tree corresponding to one part of a multi-component set creates a feature clash that necessitates the subsequent adjunction of the trees corresponding to the rest of the multi-component set.

Owen Rambow and Young-Suk Lee formalized scrambling in Korean, using multi adjunction concepts for combining a verb and its arguments in the same set (Figure 5.3), based upon the Adjoined Argument Hypothesis[44]. According to Adjoined Argument Hypothesis, all arguments are adjoined to a VP in Korean[9].

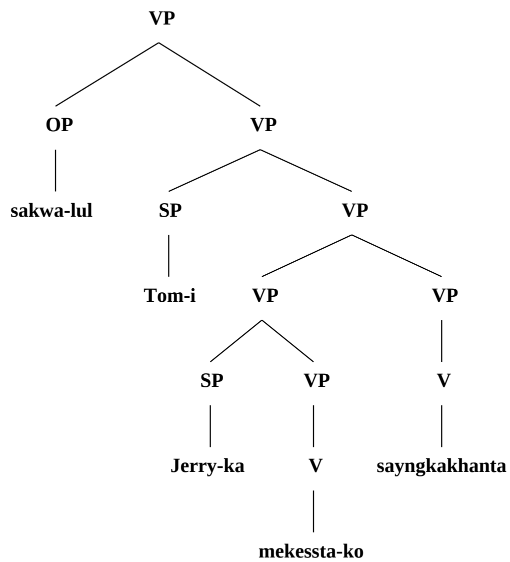


Figure 5.4: A Derived Tree by Method $\mathcal{A}$

The tree set representing a verb contains an initial tree which corresponds to the maximal projection of the verb, and auxiliary trees for each argument of the verb. As all the arguments dominate their verb in Korean, a dominance relation between the verb and the arguments is represented.

To derive sentence (27), *sakwa-lul Tom-i Jerry-ka mekess-ta-ko sayngkakha-nta*, which contains a long distance scrambled element, we start from Figure 5.3(b), which corresponds to the lexical entry for the matrix verb *sayngkakhanta* or *think*. SP$\downarrow$ is substituted with the NP argument *Tom-i*. The auxiliary tree A4 will be adjoined onto the VP node in V2. Onto the top VP node of this tree, A3 will be adjoined, again. Now, the SP node in A1 can be substituted with *Jerry-ka*, and the **OP** node in A2 can be substituted with *sakwa-lul*.

Notice that the dominance relation is only specified between V1 and A1, or V1 and A2, but

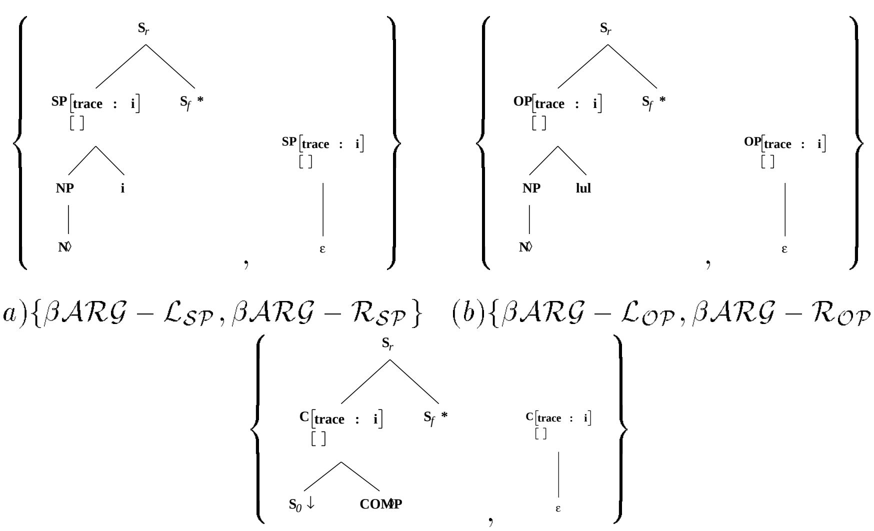
[9] In this theory, the categorial status of a clause is VP rather than IP or CP. The Adjoined Argument Hypothesis is based on the possibility that nominative/accusative case can be assigned to an adverbial, which is of the same nature as the one assigned to an argument, and that a case-marked adjunct can act as a binder for the purpose of binding.

there is no dominance between A1 and A2, in Figure 5.3(a). This allows A1 to be adjoined onto V1 before A2 being adjoined. After V1 is adjoined by A1, A2, this tree can be substituted into VP↓ node in A4. The order of operation may be different. However, as long as the dominance rule is preserved, the same derived tree as in Figure 5.4 will be drawn.

5.2.2 Method $\mathcal{B}$: A Computational Approach

The previous method (Method $\mathcal{A}$) used the concept of multi adjunction for combining a predicate and its arguments in the same set. However, there is a drawback of this method for a more computational approach for handling scrambling. For example, to know that *sakwa-lul* is an argument of the predicate *mekessta* in Figure 5.4, a derivation tree should be referenced, also. Besides, a root node of a full sentence is VP rather than S, as this method is based on the Adjoined Argument Hypothesis. Instead of using the multi adjunction concept for combining a predicate and its arguments, this concept can be used for combining a scrambled argument and its landing site.

(A) $\beta\mathcal{ARG}$ structure



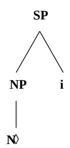
$(a)\{\beta\mathcal{ARG}-\mathcal{L}_{\mathcal{SP}}, \beta\mathcal{ARG}-\mathcal{R}_{\mathcal{SP}}\}$ $(b)\{\beta\mathcal{ARG}-\mathcal{L}_{\mathcal{OP}}, \beta\mathcal{ARG}-\mathcal{R}_{\mathcal{OP}}\}$

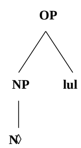
$(c)\{\beta\mathcal{ARG}-\mathcal{L}_{\mathcal{C}}, \beta\mathcal{ARG}-\mathcal{R}_{\mathcal{C}}\}$

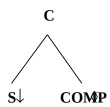
Figure 5.5: $\beta\mathcal{ARG}$ structures for Handling Scrambling

Figure 5.5(a) shows a pair of structures for representing the scrambled subject argument; the left tree represents a subject scrambled outside of its clause, and the right tree is used for representing the place where the subject should have been in the canonical sentence. Likewise, Figure 5.5(b) shows a pair of structures for representing a scrambled object argument. Figure 5.5(c) shows that a clausal argument is treated in the same way as a subject or object, i.e., as an argument.

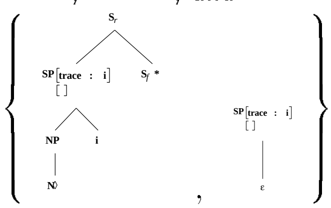
Now, we can substitute previously used elementary trees (used in Chapter 3),

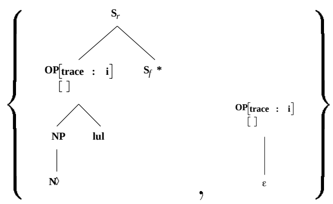


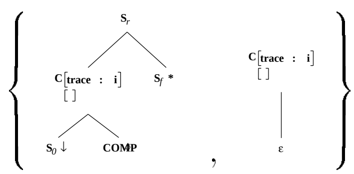
, 

, and 

with



, 

, and 

, respectively.

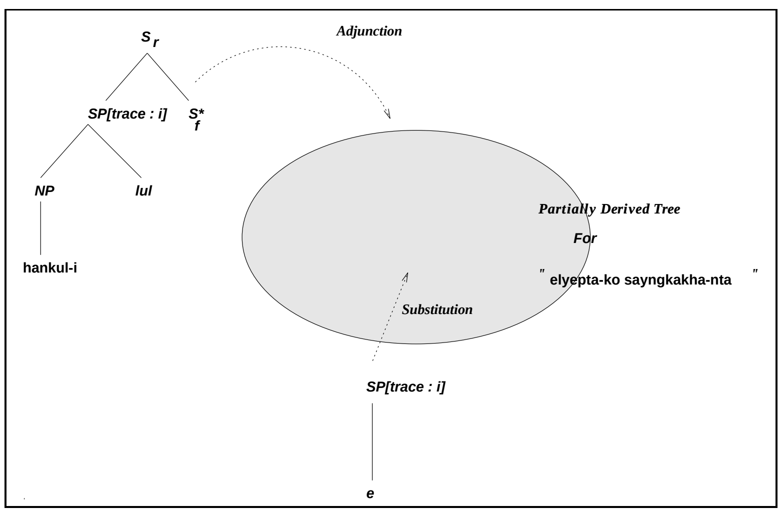


Figure 5.6: Parsing An Argument, *hankul-i*

Notice that the trace feature in each set is locally assigned the same variable for the left and right tree, so that the moved site can be correctly referenced in the final TAGs derived tree[10].

[10]For example, OP_0 and OP_1 coreference each other by the same variable of the trace feature in Figure 5.11.

For notational convenience, let's call the left tree in Figure 5.5(a) as $\beta\mathcal{ARG}-\mathcal{L}_{\mathcal{SP}}$, and the right tree as $\beta\mathcal{ARG}-\mathcal{R}_{\mathcal{SP}}$. Likewise, let's call the left tree of Figure 5.5(b) as $\beta\mathcal{ARG}-\mathcal{L}_{\mathcal{OP}}$, and the right tree as $\beta\mathcal{ARG}-\mathcal{R}_{\mathcal{OP}}$. Finally, the left tree of Figure 5.5(c) is called $\beta\mathcal{ARG}-\mathcal{L}_{\mathcal{C}}$, and the right tree, $\beta\mathcal{ARG}-\mathcal{R}_{\mathcal{C}}$.

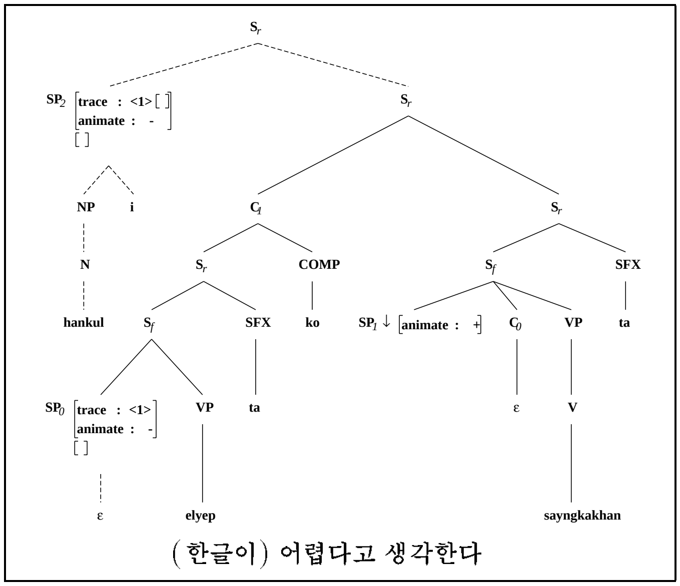


Figure 5.7: Step 1: Φ thinks that (Korean) is difficult.

Figure 5.6 sketches the operation how a subject in a scrambled position is parsed; while the $\beta\mathcal{ARG}-\mathcal{L}_{\mathcal{SP}}$ structure is adjoined onto the partially derived tree, the $\beta\mathcal{ARG}-\mathcal{R}_{\mathcal{SP}}$ structure (place holder) will be substituted into the available node in the partially derived tree. Thus, $\beta\mathcal{ARG}-\mathcal{R}_{\mathcal{SP}}$ is used for the correct indexing of the place holder.

The $\beta\mathcal{ARG}-\mathcal{R}_{\mathcal{SP}}$ structure will be dominated by $\beta\mathcal{ARG}-\mathcal{L}_{\mathcal{SP}}$, as a way of implementing a verb-final condition. Also, each S-type verb elementary tree will have a $\mathcal{NA}$ constraint on the root node, which guarantees that $\beta\mathcal{ARG}$ type structure cannot be adjoined onto any S-type structure unless its predicate structure (its S-type verb elementary tree) is already part of the partial derived tree, up to that point. As a result, a place holder structure (e.g., $\beta\mathcal{ARG}-\mathcal{R}_{\mathcal{SP}}$) cannot precede the scrambled argument structure (e.g., $\beta\mathcal{ARG}-\mathcal{L}_{\mathcal{SP}}$).

	Tom-i	hankul-i	elyep-ta-ko	sayngkakha-nta.
31	Tom-NOM	Korean-NOM	difficult-DECL-COMP	think-DECL.
	톰이	한글이	어렵다고	생각한다

For sentence (31), Figures 5.7 and 5.8 show the two steps of the derivation. Suppose that

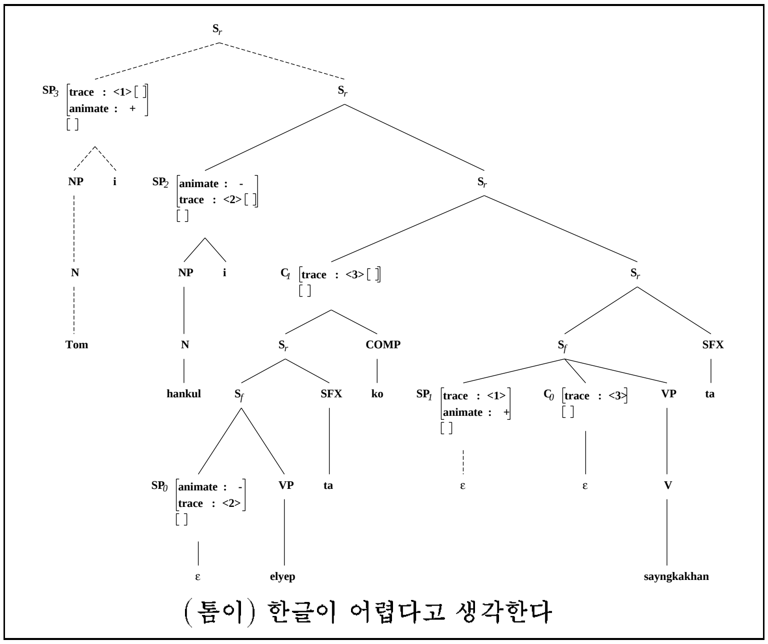


Figure 5.8: Step 2: (Tom) thinks that Korean is difficult.

we have a partial derived tree for *elyepta-ko sayngkakhan-ta* (difficult-DECL-COMP think-DECL). In Figure 5.7, $\beta\mathcal{ARG}-\mathcal{L_{SP}}$[hankul] ("$\beta\mathcal{ARG}-\mathcal{L_{SP}}$ lexicalized with *hankul*") is adjoined onto the partially derived tree, *elyepta-ko saynkakhan-ta*. Then, $\beta\mathcal{ARG}-\mathcal{R_{SP}}$ will be substituted into SP_0. As a result, the trace feature of SP_0 and SP_2 will have the same variable as each trace feature of them was locally set to equal variable, $<1>$. This enables us to represent the correct moved site for the argument (the two trees for *hankul-i* are represented with dotted lines.).

However, it can be easily noticed that SP_1 would have been a possible landing site for the substitution, as well as SP_0. In this case, $\beta\mathcal{ARG}-\mathcal{R_{SP}}$ can not be substituted into the SP_1 node as SP_1 has an [animate:+] feature. In other words, when $\beta\mathcal{ARG}-\mathcal{L_{SP}}$ is lexicalized with *hankul*, the animate feature of $\beta\mathcal{ARG}-\mathcal{R_{SP}}$ will also be set to $+$, as all the semantic features in the set will be locally set to be equal (i.e., top feature of SP node in $\beta\mathcal{ARG}-\mathcal{L_{SP}}$ $\equiv$ top feature of SP node in $\beta\mathcal{ARG}-\mathcal{R_{SP}}$), thus making it impossible for $\beta\mathcal{ARG}-\mathcal{R_{SP}}$ to be substituted into SP_1.

Figure 5.8 shows the next step. $\beta\mathcal{ARG}-\mathcal{L_{SP}}$[Tom] ("$\beta\mathcal{ARG}-\mathcal{L_{SP}}$ lexicalized with *Tom*") is adjoined onto the previous derived tree, *hankul-i elyepta-ko sayngkakhan-ta* (Korean-NOM difficult-DECL-COMP think-DECL). At the same time, $\beta\mathcal{ARG}-\mathcal{R}_{\mathcal{SP}}$ will be substituted into SP_1 node, which is the only available SP$\downarrow$ node. As it has an [animate:+] feature and $SP_1\downarrow$ also has an [animate:+] feature, it can be substituted into SP_1 node, this time.

Even if the arguments are scrambled as in *Hankul-i Tom-i elyepta-ko sayngkakha-nta* (Korean-NOM Tom-NOM difficult-DECL-COMP think-DECL), it will be correctly parsed with the correct references, as the semantic features will guide $\beta\mathcal{ARG}-\mathcal{R}$ type structures to the correct moved sites[11]. However, what happens if there is more than one possible landing site, and they cannot be distinguishable by the semantic features alone?

(B) Introduction to Priority

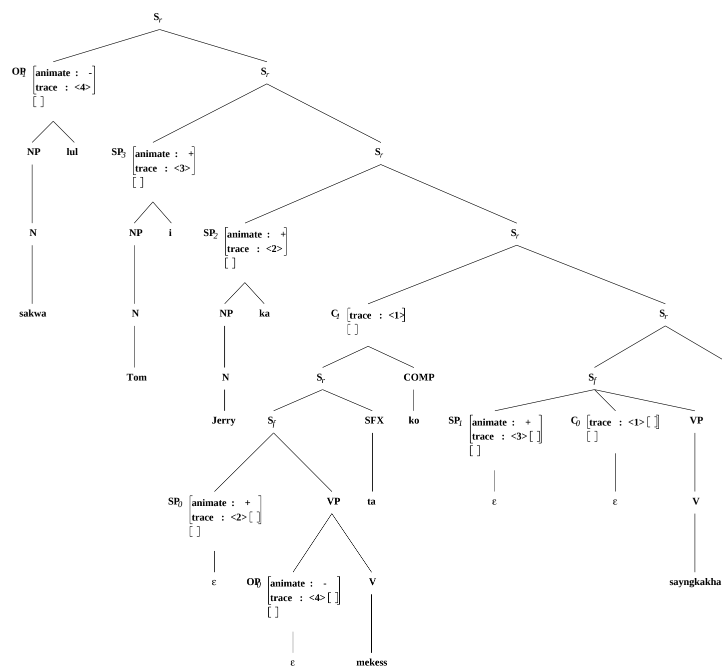


Figure 5.9: A Derived Tree by Method $\mathcal{B}$

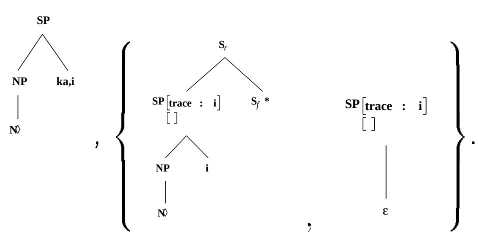
Figure 5.9 shows the final representation for the sentence (26), *sakwa-lul Tom-i Jerry-ka mekessta-ko sayngkakhanta*, using the new approach. SP_2 and SP_0, SP_3 and SP_1, and OP_1 and OP_0 are coreferenced by the trace feature attached to each node. However, with this scheme, it will not only create the Figure 5.9 tree, but also create Figure D.1 in Appendix

[11] The sentence *Hankul-i Tom-i elyepta-ko sayngkakha-nta* is awkward even though current method will be correctly parsing the sentence.

D. This is because it has two SP slots (SP_0 and SP_1) that cannot be distinguishable by the *animate* semantic feature, alone.

To resolve this problem, we need a mechanism to force the place holder tree ($\beta\mathcal{ARG}-\mathcal{R}$) to be substituted only into the closest site. For example, in Figure 5.9, when $\beta\mathcal{ARG}-\mathcal{L}_{\mathcal{SP}}$[Jerry] ("$\beta\mathcal{ARG}-\mathcal{L}_{\mathcal{SP}}$ lexicalized with *Jerry*") is adjoined onto the S-type partial derived tree, *mekessta-ko sayngkakhanta*, a substitution of $\beta\mathcal{ARG}-\mathcal{R}_{\mathcal{SP}}$ into the closest **SP** node will be tried first. In this case, SP_0 is closer than SP_1. In this way, we do not need to create Figure D.1.

Now, how the concept of Minimal Distance Rule should be implemented into XTAG system is the problem. One of the solutions might be to assign *priority* to each argument type tree. In other words, a specific structure can be given a priority over other structures[12].

Tom, N◇: 

For example, a noun, *Tom* would have two structures as above, in the Syntactic Lexicon. Let's call the first structure [$\alpha\mathcal{ARG}_{\mathcal{SP}}$ structure], and the second structure [$\beta\mathcal{ARG}_{\mathcal{SP}}$ structure][13]. Suppose the priority is given to $\alpha\mathcal{ARG}_{\mathcal{SP}}$ structure over $\beta\mathcal{ARG}_{\mathcal{SP}}$. In other words, *priority* can be given to $\alpha\mathcal{ARG}_{\mathcal{SP}}$ over $\beta\mathcal{ARG}_{\mathcal{SP}}$, to $\alpha\mathcal{ARG}_{\mathcal{OP}}$ over $\beta\mathcal{ARG}_{\mathcal{OP}}$, and to $\alpha\mathcal{ARG}_{\mathcal{C}}$ over $\beta\mathcal{ARG}_{\mathcal{C}}$.

The basic idea is that whenever an argument is not in a scrambled position, it should be substituted into an available empty slot, using the $\alpha\mathcal{ARG}$ type structure. The $\beta\mathcal{ARG}$ type structure will be used only when it is in a scrambled position, so that it cannot be subsituted into any node. Generally, when a structure is given a higher *priority* over others, and it can be successfully used for the final derivation of the sentence, the remaining structures will not be tried, at all. Only when the highest priority structure fails, the next available structure will be tried.

Actually, that the argument is in a scrambled position already implies that substitution

[12] The implementation of the priority into current TAG system is left for future work.

[13] As stated before, the left side of $\beta\mathcal{ARG}_{\mathcal{SP}}$ is called $\beta\mathcal{ARG}-\mathcal{L}_{\mathcal{SP}}$, and the right side is called $\beta\mathcal{ARG}-\mathcal{R}_{\mathcal{SP}}$.

is impossible. This conincides with our linguistic insights about Korean scrambling. In other words, this mechanism can be restated that only when $\alpha\mathcal{ARG}$ structure cannot be operated (substituted), $\beta\mathcal{ARG}$ structure will be operated (adjoined), and the correct landing site will be coreferenced by the substitution operation of the right tree of $\beta\mathcal{ARG}$ type set structure. As stated above, *priority* is always given to the $\alpha\mathcal{ARG}$ structure over the $\beta\mathcal{ARG}$.

5.2.3 An Example

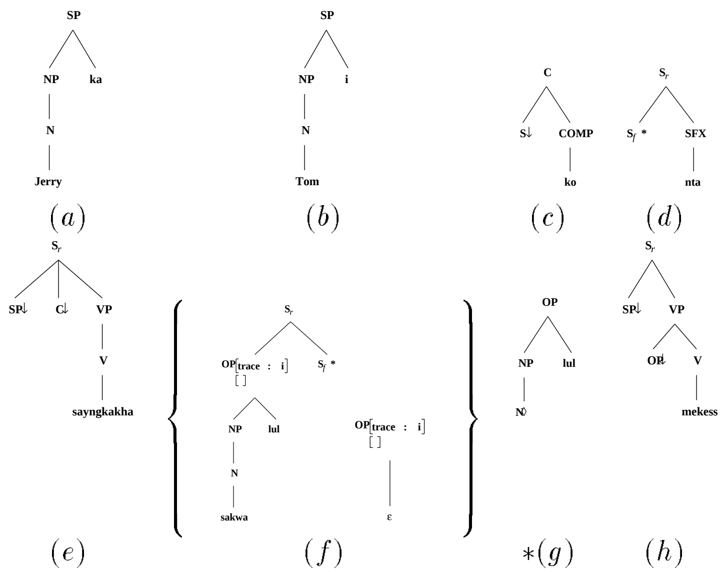
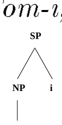
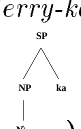
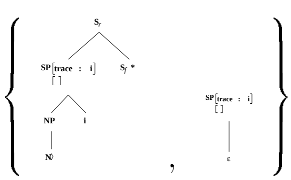


Figure 5.10: Elementary Trees

We can now derive sentence (27) from the elementary trees in Figure 5.10; there are three arguments in (27): *Tom-i, Jerry-ka,* and *sakwa-lul.* For both *Tom-i* and *Jerry-ka*, $\alpha\mathcal{ARG}$ type structure (e.g.,  or ) is used, as these two arguments are both in a canonical position. If *Tom-i* or *Jerry-ka* is in a scrambled position, then the $\beta\mathcal{ARG}$ type structure (e.g.,



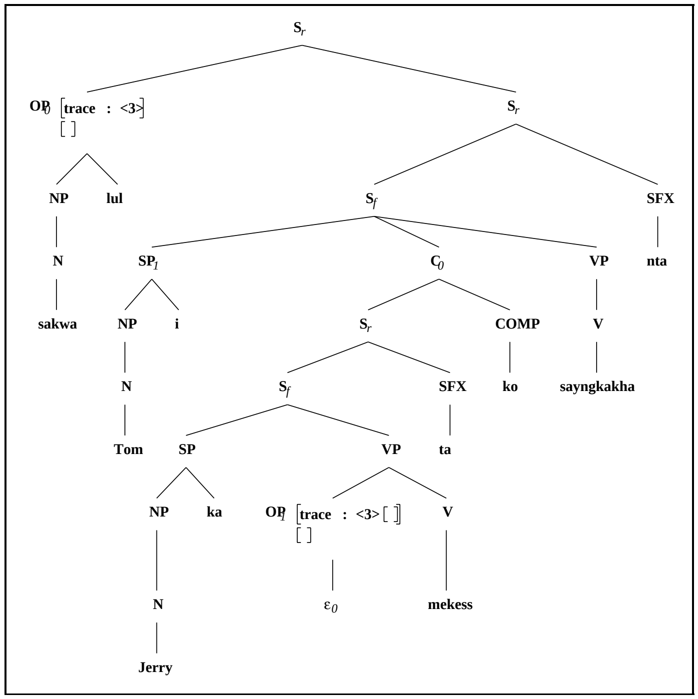
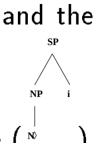


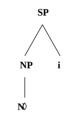
Figure 5.11: A Revised Derived Tree by Method $\mathcal{B}$

) should be used instead. In other words, as both *Tom-i* and *Jerry-ka* are not in a scrambled position and the $\alpha\mathcal{ARG}_{\mathcal{SP}}$ structure has priority over the $\beta\mathcal{ARG}_{\mathcal{SP}}$ structure, the $\alpha\mathcal{ARG}_{\mathcal{SP}}$ structure (

) must be used for both *Tom-i* and *Jerry-ka*. That is why the elementary trees in Figure 5.10(a) and 5.10(b) are used for the derivation of the sentence (27).

So, after the *mekessta-ko saynkakha-nta* partial derivation tree is created, *Jerry-ka* should be processed. As *Jerry-ka* is not in a scrambled position, $\alpha\mathcal{ARG}_{\mathcal{SP}}$[Jerry] (Figure 5.10(a)) is used to be substituted into SP node of Figure 5.11. Similarly, when *Tom-i* is parsed, $\alpha\mathcal{ARG}_{\mathcal{SP}}$[Tom] can be substituted into SP_1 node of Figure 5.11. However, *sakwa-lul* is in a scrambled position; hence, Figure 5.10(f) ($\beta\mathcal{ARG}_{\mathcal{OP}}$ structure) is used for the final derivation of sentence (27) instead of 5.10(g) ($\alpha\mathcal{ARG}_{\mathcal{OP}}$ structure). So, in Figure 5.11, $\beta\mathcal{ARG}-\mathcal{L}_{\mathcal{OP}}$ is adjoined onto the root node S_r of the partial derived tree, *Tom-i Jerry-ka mekessta-ko sayngkakhan-ta*, and $\beta\mathcal{ARG}-\mathcal{R}_{\mathcal{OP}}$ is substituted into OP_1 node. OP_0 node and OP_1 node are coreferenced by the *trace* feature.

If all the arguments in the sentence are in canonical order as in sentence (24), Figure 5.1

will be naturally derived, since only the $\alpha\mathcal{ARG}$ type structures (e.g., 

) will be used for all the three arguments.

5.3 Handling of Scrambling in Other Formalisms

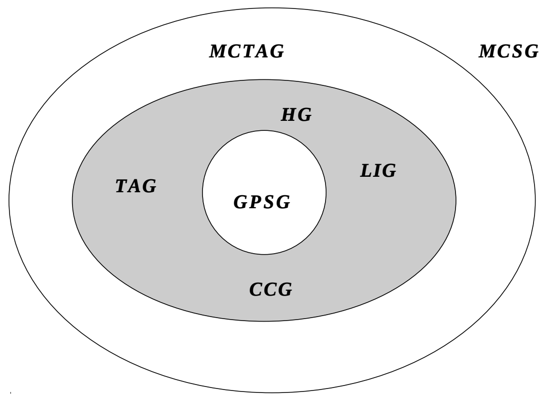


Figure 5.12: MCSG relations

Joshi showed how TAGs factor recursion and the domain of dependencies, leading to the localization of dependencies[25]. Their long-distance behavior follows from the operation of composition, which is called adjunction. He points out that TAGs have more power than CFGs, and this extra power is a corollary of factorization of recursion and the domain of dependencies. He proposed that the class of grammars that is necessary for describing natural languages be characterized as the class of *Midly Context Sensitive Grammars* (MCSGs)[14][25]. The rough properties of Mildly Context Sensitive Grammars is as follows.

" (1) Context-free languages are properly contained in MCSL.

(2) Languages in MCSL can be parsed in polynomial time.

(3) MCSGs capture only certain kinds of dependencies, such as nested dependencies and cross dependencies.

(4) MCSL have the constant growth property.[15] "[25]

[14] CCG:Combinatory Categorial Grammars,GPSG:generalized phrase structure grammars, HG:head grammars, MCTAG:multicomponent TAG, LIG: linear indexed grammars.

[15] This last property means that if the strings of a language are arranged in increasing order of length, then

Later, it was shown that Head Grammars (HGs)[42], Linear Indexed Grammars (LIGs)[16], and Combinatory Categorial Grammars (CCGs)[54] are shown to be equivalent to TAGs[57].

Here, I introduce the work done by Ik-Hwan Lee[32] for HGs and by Young-Suk Lee and Michael Niv for CCGs[35]. In these works, scrambling was handled under quite a different philosophy.

5.3.1 HGs handling of scrambling

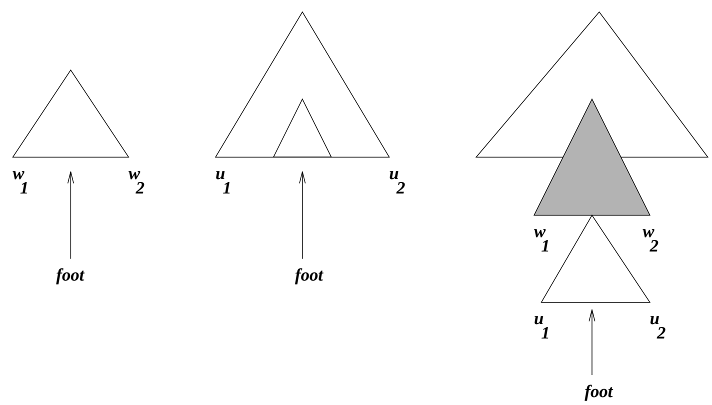


Figure 5.13: $W(w_{1\uparrow}w_2, u_{1\uparrow}u_2) = w_1u_1 \uparrow u_2w_2$

Head Grammars (HGs)[42] are a string rewriting system. But each HG string has a distinguished symbol corresponding to the head of the string. The weak equivalence of HGs and TAGs comes from a consequence of the similarities between the operations of wrapping and adjunction. Figure 5.13 shows the similarity between adjunction and wrapping using the split string notation[16].

Ik-Hwan Lee used GPSG (Generalized Phrase Structure Grammar)[16] framework with Head Wrapping Operation[17][42] for handling scrambling. Phrase structure rules are formulated as in (32)[18]. The rule format is : <rule number, syntactic rule, semantic translation>.

two consecutive lengths do not differ by arbitrarily large amounts. In fact, any given length can be described as a linear combination of a finite set of fixed lengths.

[16] Joshi points out the equivalence of the formalism using split string instead of headed string in HG

[17] HEAD WRAPPING OPERATION (HWO) RL-1 : $s_1 \ldots s_{(i-1)}w * s_i \ldots s_n$ where $s_1 \ldots s_n$ indicates the sequence of the subconstituents of a constituent.

This operation defines a syntactic function which takes two constituents w and $s_1 ... s_{(i-1)} * s_i ... s_n$. The number of '1' in RL-1 indicates the serial number of HWO's. In RL-1, the wrapping argument is the constituent depicted in $s_1 ... s_{(i-1)} * s_i ... s_n$.

[18] CM refers to Case Marker, H[1],H[2] indicate the subcategorization number of the head of IVP's. H[1] is the intransitive verb class, while H[2] designates the class of transitive verb.

	a.	<I,	$[_S$	NP,	VP],	$\alpha >$
	b.	<2,	$[_{NP}$	NP,	CM],	$\alpha >$
	c.	<3,	$[_{NP}$	Det,	N1],	$\alpha >$
32	d.	<4,	$[_{IVP}$		H[1]],	$\alpha >$
	e.	<5,	$[_{IVP}$	NP,	H[2]],	$\alpha >$
	f.	<6,	$[_{IVP}$	NP,	TVP],	$\alpha >$
	g.	<7,	$[_{IVP}$	NP,	H[2]],	$\alpha >$

The category CM denotes a function which performs an identity mapping, i.e., it takes an NP-denotation as an argument and yields an NP-denotation as its value, translated as in λ P[P].

33

Jerry-ka	sakwa-lul	mekess-ta.
Jerry-NOM	apple-ACC	eat-PAST-DECL.

With ID-rule in 32, sentence (33) is derived as in below, and translated in 34(h).

34

a. mekessta : $mekessta'$
b. Jerry : $\wedge\lambda P[P(j)]$
c. sakwa : $\wedge\lambda P[P(s)]$
d. ka : $\lambda P[P]$
e. lul : $\lambda P[P]$
f. Jerry-ka : λP[P]($\wedge\lambda$P[P(j)]) = $\wedge\lambda$P[P(m)]
g. sakwa-lul : λP[P]($\wedge\lambda$P[P(s)]) = $\wedge\lambda$P[P(s)]
h. Jerry-ka sakwa-lul mekessta : eat'($\wedge\lambda$P[P(j)])($\wedge\lambda$P[m]) = eat''(j)(s) = eat'' (j,s)

For the scrambled sentence *sakwa-lul Jerry-ka mekessta*, the subject NP Jerry-ka is the left hand argument which will get wrapped, while the IVP *sakwa-lul mekessta* (apple-ACC eat-PAST-DECL) is the wrapping argument. Accordingly, RL-1 places the subject immediately to the left of the head, *mekessta* of the wrapping argument. This operation is responsible for the derivation of the OSV type scrambled sentence. On the other hand, the canonical SOV sentence is induced by the rule $< 1, [_S NP, IVP], \Phi >$ in (32).

However, as Ik-Hwan Lee's work is only confined to local scrambling, instances of long distance scrambling should be treated by a different mechanism. In doing so, as there is a similarity between the wrapping operation in HGs and adjunction operation in TAGs, similar mechanism that was presented in the previous section can be applied to HGs.

5.3.2 CCGs handling of Scrambling

CCGs are an extension of Categorial Grammars[3], developed by Ades and Steedman[54].

Following [57], a CCG can be denoted by (V_T, V_N, S, f, R) where V_T is a finite set of terminals (lexical items), V_N is a finite set of nonterminals (atomic categories), S is a distinguished member of V_N, f is a function that maps elements of $V_t \cup \epsilon$ to finite subsets of C(V_N) is the set of categories, where $V_N \subseteq C(V_N)$ and if $c1, c2 \in C(V_N)$ then $(c_1/c_2) \in C(V_N)$ and $(c_1 \backslash c_2) \in C(V_N)$, R[19] is a finite set of combinatory rules[20].

35
ka, i : S | (S \ SP \ NP)
ul, lul : (S \ SP) | (S \ SP \ OP) \ NP
(S \ SP \ DP) | (S \ SP \ DP \ OP) \ NP
eykey : (S \ SP) | (S \ SP \ DP) \ NP

36
a. X/Y Y/Z → X/Z : $(> B)$
b. X/Y Y\Z → X\Z : $(> B_x)$
c. X/Y Y\Z → X/Z : $(< B_x)$
d. X/Y Y\Z → X\Z : $(< B)$

In TAGs, the strict structure that was related with each lexical item was an obstacle to handling scrambling. However, in CCGs, type-raised arguments can be combined together before combining with the main verb by using the idea of functional composition. The idea of functional composition is to take two functions, F and G (where F applies to the result of G) and combine them into one object before the argument for G is available: $BFG_x \equiv \lambda_x F(G_x)$. This concept was first applied for handling Korean by Lee and Niv[35]. The Korean category of particles can be represented as in (35)[21], and the $\mathcal{BLUEBIRD}$ or the Functional Composition Rules in CCGs is shown in 36 [22]. The three sentences, (24),(25), and (26) that were discussed in the previous section can be handled in CCGs as follows.

---

[19] Derivations in a CCG involve the use of the combinatory rules in R. Let the derive relation be defined as follows:
$\alpha c \beta \Rightarrow_G \alpha c_1 c_2 \beta$, if R contains a combinatory rule that has $c_1 c_2 \rightarrow c$ as an instance, and α and β are strings of categories. The string language, L(G), generated by a CCG is defined as follows:
$a_1 ... a_n | S \Rightarrow^*_G c_1 ... c_n, c_i \in f(a_i), a_i \in V_T \cup \epsilon, 1 \leq i \leq n.$
An argument of type X can be replaced by a function whose domain is the functions which map objects of type X to objects of type Y and whose codomain is objects of type Y.

[20] There are four types of combinatory rules, which involve variables x,y,z over C(V_N), and each $|_i \in \backslash, /$
1. Forward application : $(x/y)y \rightarrow x$
2. backward application: $y(x \backslash y) \rightarrow x$
3. Generalized forward composition for some $n \geq 1$: $(x/y)(...(y|_1 z_1)|_2 ...|_n Z_n) \rightarrow (...(x|_1 z_1)|_2 ...|_n z_n)$
4. Generalized backward composition for some $n \geq 1$: $(...(y|_1 z_1)|_2 ...|_n z_n)(x \backslash y) \rightarrow (...(x|_1 z_1)|_2 ...|_n z_n)$

[21] | is an abbreviation of both the forward slash and backward slash.

[22] For handling scrambled complex sentences, Lee and Niv also used Generalized Functional Composition Rules[54]

(24) :

```
Tom-i    [Jerry-ka sakwa-lul       mekessta  -ko] sayngkakhanta.
S\(S\SP) S/(S\SP)  (S\SP)/(S\SP\OP) S\SP\OP   S'\S S\SP\S'
                   ------------------------->
                          (S\SP)
         ------------------------>
                   S
                        -------------------------<
                                      S'
                                     ---------------------------<
                                               S\SP
 ----------------------------------------------->
                          S
```

(25) :

```
[Jerry-ka sakwa-lul       mekessta -ko]  Tom-i     sayngkakhanta.
S/(S\SP)  (S\SP)/(S\SP\OP) S\SP\OP  S'\S  S\(S\SP)  S\SP\S'
                                          ----------------------<B
                                                  S\S'
          ------------------------>
                   (S\SP)
------------------------>
             S
                 ----------------------<
                            S'
                          ---------------------------------------<
                                           S
```

(26) :

```
sakwa-lul        Tom-i    [Jerry-ka mekessta -ko] sayngakakhanta.
(S\SP)/(S\SP\OP) S\(S\SP) S\(S\SP)  S\SP\OP  S'\S S\SP\S'.
                          ---------------->B
                                S\OP
-----------------------<B
    S/(S\SP\OP)
                                 --------------<B
                                       S'\OP
                                      ------------------------<B2
                                              S\SP\OP
           --------------------------------------------------->
                            S
```

Recently, Hoffman is using a set notation for handling scrambling[23]. For example, a transitive verb looking for a subject and object would have the category S | {SP, OP}, and the newly defined rule is as follows:

- Forward Application (>) : $X/Y_s\ Y \rightarrow X/Y_s\text{-}\{Y\}$ where Y unifies with some $Y_i \in Y_s$.
- Backward Application (<) : $X\backslash Y_s\ Y \rightarrow X\backslash Y_s\text{-}\{Y\}$ where Y unifies with some $Y_i \in Y_s$.

5.4 Summary

The method presented here is based on the concept that from the purely syntactic point of view alone, scrambling cannot be explained. Recent study of scrambling has cast a serious doubt on the A, $\bar{A}$ explanation[23] for Japanese as well as Korean[47]. This is in contrast with explanation based on scrambling as an example of linguistic movement within the standard syntactic theories such as government and binding theory[11].

I feel that this phenomena should be explained from a broader scope, including pragmatics and other factors. Even if there is a theory to explain scrambling from a purely syntactic point of view without any confliction, it does not exclude the possibility of explaining this phenomenon with a broader scope, considering semantics or pragmatics.

Although the method used here is only using semantic features, it could be, and should be combined with pragmatics and other factors as well. This will be an important goal of my future work.

[23] e.g., government and binding theory

Chapter 6

Recovering Empty Arguments

Sentences with null arguments behave differently depending upon the characteristics of the language: they are simply ungrammatical in languages like English, whereas they are grammatical in Korean, regardless of whether the null argument appears in a subject position or an object position if the content of the null argument is recoverable from the context. These characteristics and the widespread use of empty arguments in Korean make parsing Korean text extremely problematic. Korean relies heavily on topic markers, and any argument of the verb can be omitted from a sentence, as long as it can be recovered from the context. Recently there has been an increasing amount of work in computational linguistics involving the interpretation of anaphoric elements[37], but they lack a computational component that would be useful when parsing Korean text[1].

Here, a computational method for resolving the references of empty arguments is presented that uses a stack-based discourse model and semantic features. As recovering empty arguments involves more than simple semantic features, especially when empty arguments in Korean get their referents from outside of the sentence structure, applying centering theory to a Korean discourse model is briefly discussed, which is closely related to the global recovery of empty arguments.

The problem of resolving empty arguments in Korean is closely related to the problem of matching arguments in scrambled Korean text (which was discussed in Chapter 5)[41].

[0] † This chapter has been presented as "Recovering Empty Arguments in Korean" at the 1994 Joint Conference of 8th Asian Conference on Language, Information, and Computation and the 2nd Pacific Asia Conference on Formal and Computational Linguistics, with minor modification[41].

[1] Related work for resolving English anaphora can be found in [39] and [19, 20, 53]

6.1 Topic Construction

The topic marker *-nun* is generally used in Korean to mark old information, and it precedes other information in the sentence. Once mentioned, lexical items that refer to that object may be optionally dropped as long as they can be understood from the context. In fact, any object that has been previously referred to in the discourse can be subsequently dropped from the sentence. Moon points out that a *pro* which arises by base-generating a topic with a topic marker seems to be coindexed by means of the *Minimal Distance Principle* which is an instance of the Locality Principles[2]. Consider the sentence in (37).

	Tom-un	i mwuncey-nun	phwul swu-epta ko	sayngkakhanta.
37	Tom-TOP	this problem-TOP	solve-NEG-COMP	think-PRES-DEC
	탐은	이 문제는	풀수 없다고	생각한다
	Tom$_i$ thinks that	he$_i$ can not	solve this problem.	

The two noun phrases in the sentence: *Tom* and *i mwuncey* are both marked with the topic marker *-nun*[3]. There are a couple of ways to interpret the use of the topic marker in this sentence. One way is to consider the topic marker *-nun* as simply ambiguous between being a subject or object marker. The *un* in Tom-un would be considered a subject marker, while the *nun* in mwuncey-nun would be considered an object marker. This is unappealing from a computational point of view, since it provides no help in parsing the sentence or resolving possible ambiguities.

Another method is to consider the arguments of the two verbs, *phwul swu-epta* and *sayngkakhanta*, to be empty, with the topic markers optionally adjoined onto the beginning of the sentence. The references for the empty arguments must be obtained from the earlier context of the sentence, i.e. the topic NPs. A graphical representation of this way of viewing the sentence is given in Figure 6.1. The topic noun phrases (TP) are at the beginning of the sentence, while the subject argument for *sayngkakhanta* and the subject and object arguments for *phwul swu-epta*[4] are all empty.

[2] He claims that the *Minimal Distance Principle* is related to the asymmetry between the subject pro and the object pro which indicates that the subject pro cannot be moved, whereas the object pro can be moved to the adjacent position of the topic[36]. His claim that there is no asymmetry between lexical subject and lexical object with respect to movement contradicts Saito[45]'s generalization concerning the subject-object asymmetry in Japanese in respect to movement.

[3] Because of phonological considerations, the *-nun* marker becomes *-un* after an *[m]*.

[4] For notational convenience, *swu* and *epta* are treated together as an auxiliary verb.

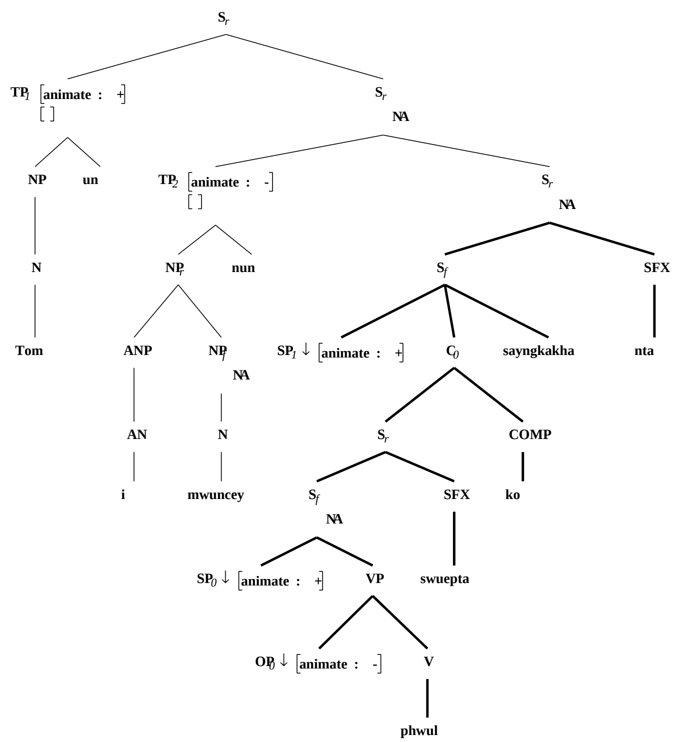


Figure 6.1: Tom-un i mwuncey-nun phwul swu-ep-ta ko sayngkakha-nta

It is also possible for the topic noun phrases to have been referenced previously in the discourse, and not show up explicitly in the sentence at all. In this case, the references for the empty arguments must be picked up from the wider context. This provides an even stronger motivation for the analysis in which the arguments are empty and the topic optionally adjoins on. The ambiguous-null method must have a separate mechanism for handling sentences in which there are no topic markers within the sentence itself. With the empty-argument method (as shown in Figure 6.1), the arguments are empty whether the topicalized NP is in the sentence or not. This means that a single mechanism can be utilized to resolve the references, since their meaning comes from a context outside of the argument scope of the verb.

The constraints on long-distance scrambling in Korean and the techniques used for recovering elided noun phrases in telegraphic English[39], and for resolving English anaphora[19, 20, 53] provide some insight into a possible computational mechanism to recover missing arguments of a verb.

Long-distance scrambling provides evidence that the verb looks to the topic closest to it. Previously, we saw in sentences (27) and (28) that when *Jerry* and *Tom* are scrambled so that *Tom* is closer to the verb *mekess* or *ate*, *Tom* becomes its subject, leaving *Jerry* as the subject of the verb *sayngkakhanta* (*think*).

6.2 Applying Semantic Features

6.2.1 A General Algorithm

The problems of resolving the moved arguments in scrambling and recovering empty arguments are closely related. Both problems can be viewed as a need to find the argument of a verb that is not where one might expect it to be, either because it has scrambled out of position, or because it has been dropped from the sentence.

We (H.S. Park, Dania Egedi, Martha Palmer) propose a general rule for recovering the referent for empty arguments as follows: Choose the closest topic that matches the semantic constraints on the elided arguments. This would most easily be implemented with a stack mechanism that pushes new topics onto the stack as they are encountered. As each topic is encountered in the sentence, it is pushed onto a stack, along with the semantic features associated with the lexical item.

6.2.2 An Example

Consider the sentence in (37) again. Its graphical representation was given in Figure 6.1, and showed the empty arguments for the verbs *sayngkakhanta* (*think*) and *phwul swuepta* (*can't solve*), with the topic NPs adjoined onto the beginning of the sentence.

Figure 6.2c shows the state of the stack after both topic NPs have been pushed onto it. After processing the topic NPs in the sentence, we come to the empty subject of *sayngkakhanta* (*think*). Figure 6.2a shows the semantic feature constraints on the base tree associated with *sayngkakhanta* (*think*). The subject argument is constrained to be [animate:+].

The argument closest to the verb (OP_0) is processed, first. The object argument is constrained to be [animate-], and looking at the the stack, the top NP is *mwuncey* (*problem*), which is [animate-], so it fills that argument. *Mwuncey* is then popped off the stack, leaving

Tom on the top of the stack. The next empty argument is the subject of *phwul*, which we will skip over here since we believe that it is actually an instance of PRO-control[5]. The next empty argument is the subject of the main verb *sayngkakha* (*think*), whose tree is given in Figure 6.2c. The subject is constrained to be [animate+]. The top NP on the stack (*Tom*) is also [animate+], so it can fill the subject slot for *sayngkakha*.

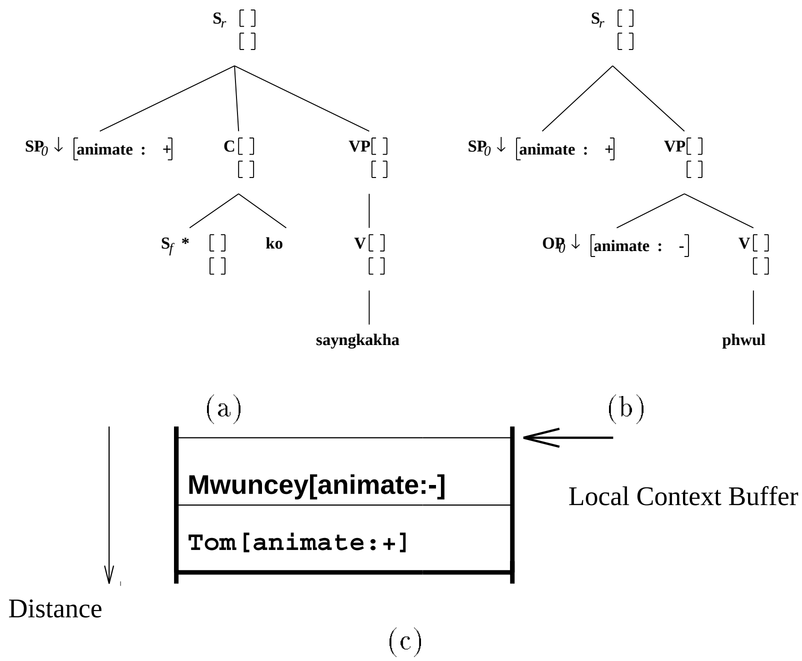


Figure 6.2: Recovering empty arguments using a stack

Note that the above example would work just as well without the semantic features, since the topicalized noun phrases are in the canonical order. Simply popping them off the stack would be enough. However, if the topicalized NPs were in a different order, as in sentence (38), then the semantic features, along with the look-ahead capabilities of the local stack, would be

[5]There are strong arguments for the subject of the embedded clause to be an instance of PRO-control [11], and as such it would get its referent from the subject of the matrix clause. We do not want to get into those arguments here, as it is certainly possible for the subject of the embedded clause to undergo the argument recovery process as well, if one wanted to argue against a PRO control analysis. The sentence, of course, would not have the correct number of arguments, and would need to select one of its arguments from the global ordered list. Since the object NP of the embedded clause has already been filled by *mwuncey*, the subject of the embedded clause, which is constrained to be [animate+] would try to match with the current top NP on the local stack, *Tom*. Their features are compatible, so *Tom* would be popped off the local stack and added to the global ordered list. The last NP argument to be filled, the subject position of *sayngkakhanta* (*think*), would then look to the global ordered list, and select *Tom* as the most pertinent, compatible NP. We do not yet have an algorithm for ordering the NPs in the global list, but in this particular case, one would imagine that the last things mentioned would be at the top of the list.

needed to correctly fill the empty arguments.

38				
	i mwuncey-nun	Tom-un	phwul swu-epta ko	sayngkakhanta.
	this problem-TOP	Tom-TOP	solve-NEG-COMP	think-PRES-DEC
	이 문제는	탐은	풀수 없다고	생각한다
	Tom_i thinks that	he_i can not	solve this problem.	

In this case, *Tom* would be on the top of the stack. The object NP of *phwul* (*solve*) would first try to match with *Tom*, but the semantic features are incompatible. It would then look-ahead on the stack, until it came to an NP that had compatible features, in this case, *mwuncey*. *Mwuncey* would then be popped off the stack, and the algorithm would continue.

While it is not necessary for all of the missing arguments to be available within the sentence itself (since additional topics may be taken from the discourse list), it is necessary for all of the NPs in a sentence to fill an argument within that sentence. The use of a local stack helps guarantee this. Consider the sentence in (39).

39				
	*Tom-un	Jerry-nun	Mary-nun	coaha-nta.
	Tom-TOP	Jerry-TOP	Mary-TOP	like-PRES-DEC

Coahanta (*like*) takes two arguments - a subject and an object NP. When processing the sentence, all three NPs are placed onto the stack. *Mary* would resolve the object argument, and be popped off the stack. *Jerry* would then resolve the subject argument, and be popped off the stack as well. This leaves *Tom* on the stack with no role left in the sentence to be assigned to it. This would cause the sentence to fail, as it should.

6.2.3 Recovery at Discourse Level

40			
	ku pokose-lul	pwunsilhayssta-ko	malhayssta.
	that report-ACC	lose-PAST-DEC-COMP	say-PAST-DEC.
	그 보고서를	분실했다고	말했다
	Φ said that Φ lost	that report.	

In the previous example (37), all the arguments for filling the empty slots were provided within the sentential level, even though the mechanism of a global list was used. However, it is also possible that the empty arguments be provided in the context. For example, sentence (40) is ungrammatical at the sentential level as the empty arguments for two subjects cannot be recovered at all. However, if sentence (41) was the previous utterance before sentence (40), it is obvious that the empty arguments should be filled by *Tom*. Even when the arguments are recovered from the context, the semantic features will have to match to fill the empty slots.

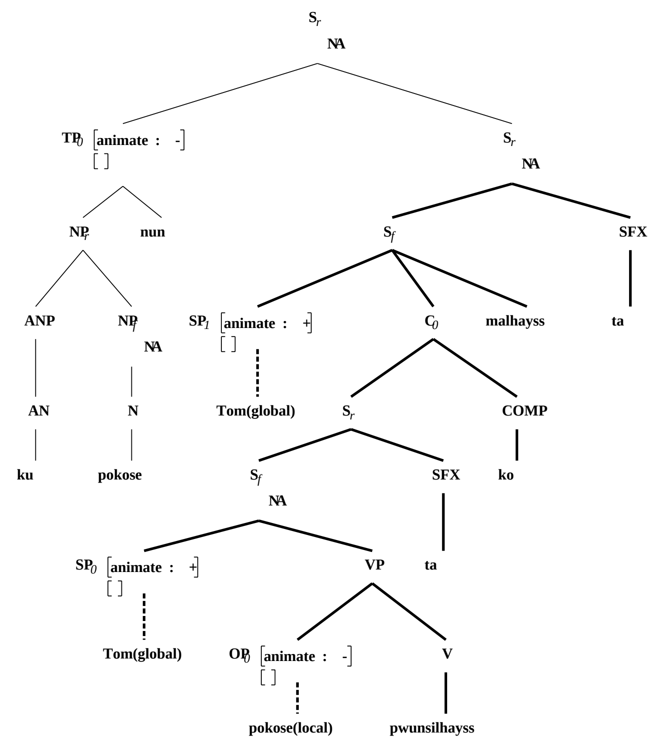


Figure 6.3: 보고서는 분실했다고 말했다 (*(Tom) said (he) lost that report*)

	Tom-i	ku pokose-ey kwanhay	mwuela-ko	malhayssni?
41	Tom-NOM	that report-LOC	what-COMP	say-PAST-INT
	톰이	그 보고서에 관해	무어라고	말했니?
	What	did Tom say	about	that report?

6.2.4 A Counter Example

In addition to the problem of deciding how the discourse list should be ordered (a very hard problem in and of itself[12]), the interaction between the local stack and the global discourse list is not clear to us. Consider the example in sentence (42)[6].

	Tom-i	[Mary-ka	ϵ	hyeppakha-yssta-ko]	malha-yssta
42	톰이	메리가	ϵ	협박했다고	말했다
	Tom-NOM	Mary-NOM	EC	threaten-PAST-COMP	say-PAST-DECL
	Tom	said	that Mary	threatened	ϵ

According to the algorithm given in the previous subsection 6.2, the object argument of *hyeppakhaysstа* (*threatened*) should be filled by *Mary*. This, however, is not the case. It is not

[6]Thanks to Bonnie Dorr for pointing out this apparent counter-example to our general algorithm.

correct either, though, to assume that it will be bound to *Tom*, the other NP in the sentence. There are not enough NPs in the sentence to fill all of the available empty arguments, and so one argument must be filled from context (which may be *Tom* or some other NP). We believe that considerations such as pragmatics and discourse theory, as well as semantics, play a role in deciding which empty arguments are bound outside of the sentence. The fact that ϵ is unlikely to be bound to *Mary* is most likely based on pragmatic considerations, such as the fact that one does not usually threaten oneself. Given a context for sentence (42), the empty argument could be filled either by *Tom*, or by something available in the context. The situation in (43) shows a context in which the empty argument in the second sentence would be resolved with *Jerry*, not *Tom*.

	Jerry-ka			hyeppaktanghayssta.	
	제리가			협박당했다	
	Jerry-NOM			threaten-PASSIVE	
	Jerry			was threatened.	
43					
	Tom-i	[Mary-ka	ϵ	hyeppakha-yssta-ko]	malha-yssta.
	톰이	메리가	*epsilon*	협박했다고	말했다.
	Tom-NOM	Mary-NOM	EC	threaten-PAST-COMP	say-PAST-DECL
	Tom	said that	Mary	threatened	Jerry.

6.3 Applying Centering Theory

The previous section dealt with recovering arguments based on semantic features. However, it is obvious that there will be numerous other factors that affect the recovery of empty arguments. Even though studying all the possible factors would be beyond the scope of this paper, I would like to introduce the centering theory which was originally discussed by Grosz[17], and briefly discuss how it can be applied to the recovery of empty arguments. This is based on Kameyama's work on Japanese[28].

6.3.1 Centering Theory

Centering theory is originally introduced by Grosz[17]. Later, Kameyama generalized it and applied it to discourse processing in Japanese[28].

Within the frame work of centering theory, each utterance in a discourse model is associated with a set of discourse entities called Forward-Looking Centers C_f. Among this set, there is a special member called the Backward-Looking Center. The C_b entity links the current utterance to the previous discourse. The set of Forward-Looking Centers are ranked according to discourse salience. The highest ranked member of the set is called as the *Preferred Center* C_p. In addition to $C_p, C_b, and C_f$, the theory specifies a set of rules and constraints[28].

CONSTRAINTS : For each utterance U_i in a discourse segment $U_1, ..., U_m$:

1. There is precisely one backward looking center C_b.
2. Every element of the forward centers list, $C_f(U_i)$, must be realized in U_i.
3. The center, C_b, is the highest-ranked element of $C_f(U_{i-1})$.

It provided a study of syntactic factors in Japanese on discourse interpretation, and it was further developed by Walker et al[58]. Kameyama's CENTER RETENTIONAL RULE and CENTER ESTABLISHMENT RULE for Japanese seem to be working almost identically for Korean, also.

The CENTER RETENTIONAL RULE for Korean : If the C_b of the current utterance is the same as the C_b of the previous utterance, a zero pronominal should be used.

The CENTER ESTABLISHMENT RULE for Korean : If one of the C_f in the previous utterance is made into the C_b of the current utterance, a zero pronominal is used.

In addition to constraints and rules, the modeling of attentional states depends on analyzing adjacent utterances in a discourse according to a set of transitions. According to Walker et al., the transition from one utterance, U_{i-1} to U_i is based on :

(a) whether the backward-looking center, C_b is the same from U_{i-1} to U_i,

or

(b) whether the discourse entity is the same as the preferred center, C_p.

If both hold, then it is a CONTINUE transition. If (a) holds, but not (b), it is a RETAIN transition. If (a) does not hold, it is a SHIFT transition[58].

	$C_b(U_i) = C_b(U_{i-1})$	$C_b(U_i) \neq C_b(U_{i-1})$
$C_b(U_i) = C_p(U_i)$	CONTINUE	SMOOTH-SHIFT
$C_b(U_i) \neq C_p(U_i)$	RETAIN	SHIFT

Other constraints that are lexically specified such as [animate:+] can be easily applied to the centering theory.

6.3.2 An Example

	(a)	Lee-ka	say pokose-lul	chaksenghayssta.
		Lee-NOM	new report-ACC	make-PAST-DECL
		Lee made	a new report	
	(b)	Kim-eykey	pokose-lul	poyecwuessta.
44		Kim-DAT	report-ACC	show-PAST-DECL
		(He) showed	the report to Kim	
	(c)	(kuliko)	nayyong-ul	selmyenghayssta.
		(and)	content-ACC	explain-PAST-DECL
		and (he) explained	the content	

Here is the example of applying centering theory for recovering empty arguments. In 44, the C_b of sentence (44)(a) is [Lee], and C_f is [Lee, Report]. In sentence (44)(b), C_b is still [Lee], and C_f is [Lee, Kim, Report]. As it satisfies the condition $C_b(U_i) = C_b(U_{i-1})$ and $C_b(U_i){=}C_p(U_i)$, it is a CONTINUE transition.

In sentence (44)(c), whereas C_b is still Lee, C_f can be either [Lee, Kim] or [Kim, Lee], thus making two possible transitions; CONTINUE, or RETAIN. However, [Lee] is strongly preferred against [Kim] as C_p. The reason is because [Lee] was the subject and [Kim] was just the indirect Object. There is a scale of relative salience. Following Kameyama, I assume the following C_f rankings for Korean : $\text{Topic} > \text{Empathy} > \text{Subj} > \text{Obj2} > \text{Obj} > \text{Others}$.

This shows that the case marker alone can be an important factor that has an influence on recovering empty arguments in the discourse model. However, even in applying centering theory, sometimes pragmatics overrides the effect of centering.

6.4 Summary

The prevalence of empty arguments in Korean makes it a virtual necessity for any Korean parser to be able to recover the referent of the missing argument if there is to be any hope of providing the most rudimentary understanding (or translation) of the sentence. We (H.S. Park, D. Egedi, M. Palmer) have identified a stack-based, computational algorithm for resolving empty arguments that is based on semantic constraints motivated by the constraints found in scrambling. Other than semantic features, applying centering theory is discussed. As far as the recovery of empty arguments is concerned, semantic feature constraints seemed to work for the discourse level as well, and could be applied together with centering theory. At this moment, though, how this formula as well as other factors should be combined together in a more formal way needs further research.

Although the basic recovery algorithm seems to work for recovering arguments at the discourse level, the interaction between local list and global context should be further investigated[7].

The research on empty argument recovery in a discourse model will be used in a machine translation system between English and Korean using Synchronous Tree Adjoining Grammars [51]. In addition, we (H.S. Park, D.Egedi, M.Palmer) are investigating the interaction between empty argument recovery and PRO-control in Korean.

The current study of centering is concerned with the analysis of a discourse from a linguistic structure of discourse itself. As a discourse should be related to the study of the cognitive processes necessary for generating, as well as the study of interpreting the text, further studies will be focused on a wider spectrum of areas such as applying centering theory to an actual computational system. The research on how centering theory[28, 18] could be used in conjunction with the global ordered list to choose the correct referent for empty arguments would be one of the most challenging ones.

[7] Work on implementing this empty argument recovery algorithm is experimented using a Combinatory Categorial Grammar (CCG)[54] parser written in Prolog. See Appendix E for the code.

Chapter 7

Conclusion and Future Work

Most of Korean lexical structures have been expressed in TAGs. Even though there were several problematic structures which still need to be addressed, it was not due to the lack of power of TAG formalization, but rather due to controversial linguistic theory of Korean. As a matter of fact, in applying to U.S. Army Telecommunication Messages, some of the TAG representations look awkward from the linguistic point of view.

However, it should be remembered that TAG here is not used for linguistic representation. It is purely a way of showing a parsing method, and tries to preserve the concept of domain of locality in every Korean lexical element, which might be eventually a good way of representing Korean grammar.

I also presented a prototype Synchronous TAGs transfer based system, augmented with semantic feature unification. This system is being applied to a domain of military messages for translation between Korean and English. Augmenting this method in several areas such as pronoun reference, situation representation is being studied.

I also discussed how to handle scrambling using MC-TAGs. In doing so, the priority concept was newly introduced for the proper handling of scrambling. I tried to interpret the scrambling phenomenon with regard to a wider range of factors such as semantic features and pragmatics. Actually, the approach to scrambling was based on the idea that scrambling could freely occur as long as it did not make a sentence ambiguous. In this paper, only the semantic features for handling scrambling were focussed on. Further research needs to be done on the implementation as well as on the nature of scrambling itself.

As empty arguments were so prevalent in our telecommunication messages, I presented a computational algorithm. As this topic is directly related to a discourse model, the algorithm needs to be augmented with other discourse oriented theories such as centering.

Finally, the work presented here is far from perfect. However, I hope this will contribute to further research in this area for me and other researchers.

Appendix A

US Military Telecommunication Messages

REQUEST STATUS ON CLASS 1 ITEMS 103 FSB.
1급 조항 103 전위지원대대에 관한 상황 요청

WE CURRENTLY SHOW ZERO STATUS ON COMMANDERS REPORT.
현재 우리는 지휘관 보고 현황에 관한 한 영위에 있다.

LTC LEEH, XO DISCOM.
LEEH 육군중령, 사단지원사령부 행정관.

PLEASE UPDATE 103 FSB, AND SEND IT TO THIS LOCATION.
103 전위지원대대에 관한 최근 정보를 수집하여, 그것을 이곳으로 보내 달라.

NEED AN UPDATED COMMANDERS REPORT.
최근의 지휘관 보고가 필요하다.

SPEEDY, CAN WE GET A COMM REP ON ALL BDE?
Speedy, 모든 여단의 (지휘관 보고 — 통신 응답)를 얻고 싶다.

WAS THIS PROBLEM SELF-INFLICTED?
이 문제는 자생적인 것인가?

SEND REPORT AGAIN, HAVING TROUBLE ENTERING THE DATA BASE.
보고를 다시 해 달라. 데이타베이스에 문제가 생겼다.

COMMO IS UP. PLEASE SEND A CURRENT CMDRS REPORT ON ALL UNITS.
통신이 이제 가능하다. 모든 부대에 현재 지휘관 보고를 보내라.

LOST YOUR REPORT. PLEASE SEND AGAIN.
보고를 분실했다. 다시 보내라.

**ALSO, REQUEST PROPOSED LOC & ESTIMATED TIME DISCOM WILL MOVE TO THE NEW LOCATIONS.
**예정된 장소와 사단 지원 사령부가 새 장소로 이동하는 대략의 시간을 알려달라[1].

[0]† This data is supplied by Mr. Yaeger, US army research officer, and translated by Sungki Suh and Jeyhoon Lee, at the University of Maryland. Some of the translation might not be appropriate for the actual military messages.

[1]This message has a conjunctive structure of a simple noun phrase and a noun phrase modified by a relative clause.

DID YOU RECEIVE OUR LAST REQUEST? ****WE NEED AN UPDATED OR CURRENT CMDRS REPORT ASAP.
우리의 최근 요청을 수신했는가? ****우리는 최근 혹은 현재의 지휘관 보고 내용을 가능한한 빨리 필요하다[2].

THE ONLY UNIT WE HAVE A CMDR REPORT ON IS THE 149, WE ARE NOT IN CONTACT VIA MCS WITH THE 67, 69 OR 1-167.
지휘관 보고가 입수된 유일한 부대는 149이다. 67, 69, 그리고 1-167 과는 기동제어시스템을 통한 접촉이 되고 있지 않다.

****WHO IS IN CONTACT WITH 67, 69, AND THE 1-167 IF YOU AREN'T?
****67, 69, 그리고 1-167 과 접촉을 귀하가 아니라면 누가 하고 있는 것인가?[3]

WE'VE TALKED TO THEM ON THE PHONE TO GET UNIT LOCATIONS, BUT BEYOND THAT WE HAVE NOTHING ON THEM.
우리는 부대 위치를 알기 위해 그들과 전화로 교신했다. 하지만 그 이상은 한 일이 없다.

REFERENCE TO YOUR REQUEST FOR CURRENT CMDS REPORT. WILL SEND BY 0600.
귀하의 현재 지휘관 보고 요청에 관한 참고 자료이다. 0600까지 보내주겠다.

WE ARE IN THE PROCESS OF FULFILLING YOUR REQUEST. WILL SEND WHEN COMPLETED.
우리는 현재 귀하의 요청을 수행하고 있는 중이다. 완료되는 대로 보내 주겠다.

CONTACT BY LAN WITH DIVARTY ACCOMPLISHED AT 0706.
지역통신망으로 0706에 완수한 사단 포병대와 접촉하라.

PLEASE UPDATE CMDR'S REPORT FOR 230600FE91.
230600FE91에 관한 최근 지휘관 보고를 알려달라.

DO YOU HAVE ANY INFO FOR US YET?
우리에게 알려 줄 어떤 정보라도 갖고 있는가?

IF YOU HAVE TROUBLE RETRIEVING THE IBP FILE WE JUST SENT, PLEASE CONTACT YOUR MC REP. FOR ASSISTANCE.
우리가 방금 보낸 IBP화일을 검색하는데 어려움이 생기면 MC REP와 접촉하여 도움을 얻으라.

WHAT ACTIONS, IF ANY, ARE REQUIRED?
필요하다면, 어떤 행동을 취해야 하는가?

PLEASE RESPOND TO 35 DISCOM IF YOU HAVE RECEIVED THIS MESSAGE AT NODE CD.
귀하가 이 소식을 NODE CD에서 수신했다면, 35 사단지원사령부에 응답하라.

PLEASE SEND PREVIOUS QUERY TO 35 DIVARTY CAN NOT REACH ON LAN.
이전의 의문사항을 지역통신망으로는 교신 불가능한 35 사단포병대에 보내라.

HAVE RECEIVED ACK FROM YOUR NODE. HAVE NOT RECEIVED ACK OR MSG ABORT FROM 35 G-3 OPS. IS CS OPERATIONAL AT G-3?
귀하의 NODE로부터의 접수를 통보받았다. 35 G-3 작전에 관해선 접수 통보나 기타 연락을 받지 못했음. G-3 에서 회로 스위치가 작동하는가?

CPT JACKSON

2

우리는	최근	혹은	현재의	지휘관 보고 내용을	가능한	한	빨리	필요하다
We-TOP	updated	or	current	CMDRS-ACC??	possible-SFX	DNP	fast	need-DECL

This message could not be handled as *pilyohata* (필요하다) is an intransitive verb, and accusitive markers are used for *nayong-ul*. If *nayyong-ul* is changed to *nayyong-i*, it can be parsed. At this moment, I am not sure whether it is a general human mistake that can be still understood by a native speaker, or it is a problem of case overlapping.

3

67, 69, 그리고 1-167 과	접촉을	귀하가	아니라면	누가	하고	있는	것인가?
67, 69, and 1-67-PTC	contact-ACC	[you	NEG-SFX]	who-NOM	do	ing	BE-INTR

This message could not be handled because it involved long-distance scrambling as well as local scrambling. In other words, 접촉을 (contact-ACC) and 누가 (who-NOM) are locally scrambled, and the clause itself is inserted into the main sentence.

대위 JACKSON

MCS TEAM LEAD
기동제어시스템팀장

CMDR REPORT FOR THE 6 BDE.
6 여단 지휘관 보고.

69TH, WE GOT YOUR MESSAGE CMDR REPORT FOR THE 69 BDE , BUT WE DIDN'T GET THE ACTUAL REPORT, EITHER TRY AGAIN OR TURN ON YOUR AUTO LIGHT.
귀하로부터의 통보- 69여단 지휘관 보고 를 받았다. 그러나 실질적인 보고를 받지 못했음. 다시 보내 주거나 아니면 귀하의 AUTO LIGHT 를 켤 것.

SEND US A CMDR'S REPORT.
지휘관 보고를 보내 달라.

REQUEST COMMANDER REPORT CURRENT FROM DIVARTY UNITS WITHIN YOUR AREA.
귀하 지역의 사단포병 부대들로부터의 현재 지휘관 보고를 요청한다.

REQUEST ALL NBC REPORTS.
모든 화생방 보고를 요청한다.

LAST MSG?
최근 통보는?

FSE NEED A GUMBALL REPORT ON ALL DIVARTY UNITS.
화재지원편대는 모든 사단포병대에 관한 Gumball 보고를 필요로 한다.

ON THE RADIO I HEARD SOMEONE AT NB 440191. WHO IS THAT?
라디오 상에서 NB 440191의 누군가의 목소리를 들었다. 누구인가?

DID THE BDE CDR RETIRE THE FLAG FOR 2/635 AR? REQUEST FROM PREVIOUS BN CDR FOR THE FLAG HAS BEEN SUBMITTED.
여단 지휘관이 2/635 기갑부대의 기를 철회시켰는가? 이전 대대 지휘관으로부터 기를 제출해달라는 요청이 있음.

COULD YOU CHECK?
확인해 줄 수 있는가?[4]

****NEED VERIFICATION ON MESSAGE FROM X CORP ABOUT INFO FROM CEWI BEING BOGUS ABOUT 79 TD WILL NOT HIT IGB UNTIL MID-NIGHT.
****자정까지는 79 탱크 사단이 독일 국경을 공격하지 않을 것이라는 통신전 정보부로부터의 정보가 허위라는 X CORP로부터의 통신을 확인하고 싶다[5].

PLEASE SEND YOUR LATEST COMMANDER'S REPORT.
최근의 귀하 지휘관 보고를 보내 달라.

RECEIVE BY SPC HOKE, 1914 23FEB91.
SPC Hoke 에 의해 1991년 2월 23일 19시 14분 수신.

PLEASE SEND CURRENT LOCATIONS OF BNS. THANKS FOR YOUR SUPPORT.
대대의 현재 위치를 알려달라. 지원해 주어서 고맙다.

DO YOU HAVE THE CURRENT FRONT LINE TRACE? IF SO PLEASE SEND.
전위 부대의 흔적을 파악하고 있는가? 알고 있으면 우리에게 알려달라.

DID YOU GET LAST MESSAGE? IF NOT HERE IS A REPEAT. WE WANT TO KNOW IF YOU HAVE THE CURRENT FRONT LINE TRACE IF SO PLEASE SEND IT.

---

[4] The word *swu* (수) is treated as a noun phrase with the missing subject case marker.

[5] There are three places where *nun* (는) is appearing: *caceng-kkaci-nun* (by noon-nun), *ila-nun* (BE-nun), and *hewila-nun* (BOGUS-be-NUN). The second and third *nun* are just adnominal suffixes. However, this message could not be handled as the structure of the first lexical item *nun* (는) in *caceng-kkaci-nun* (자정까지-는) could not be analyzed, at this moment.

방금 통신된 것을 수신했는가? 수신 못 했다면 지금 다시 보내 주겠다. 귀하가 전위부대의 흔적을 파악하고 있는지 알고 싶다. 만약 알고 있으면 우리에게 알려달라.

REQUEST GRID OF ALTERNATE POSITION FOR CHEMICAL MUNITIONS ASAP. WE DO NOT HAVE EARLIER MESSAGE LISTING GRID.
가능한 한 빨리 화학 군수품 교류 grid을 요청한다. 지난번 메시지를 열거하고 있는 grid를 우리는 가지고 있지 않다.

WHAT ARE YOUR CURRENT LOCATIONS FOR 149 BDE? ARE THEY MOVING?
149여단의 현재 위치가 어디라고 생각하는가? 그들이 이동중인가?

PLEASE SEND NEW LOCATIONS OF MLRS PLATOONS OF 2-675.
2-675 다발 로켓 시스템 소대의 위치가 어디인지 알려달라.

NEED TIME AND LOCATION FOR CLASS IV CCL I ASAP.
가능한 한 빨리 class IV CCL I 의 시간과 위치를 알고 싶다.

SEND PRESENT LOCATION AND DIRECTION AND SPEED OF MOVEMENT, IF ANY.
가능하다면 현재 위치와 이동 방향 그리고 이동 속도를 알려 달라.

PLEASE SUBMIT CENTER OF MASS FOR HQ 69 BDE, HQ 67 BDE, HQ 149 BDE.
69 여단 본부, 67여단 본부, 그리고 149 여단 본부의 center of mass를 제출해 달라.

PLEASE SEND GRID CORD. OF CENTER MASS OF DE. AND GIVE DIRECTION AND SPEED OF MOVEMENT. ACKNOWLEDGE WHEN RECEIVED.
DE. 의 center mass 의 Grid CORD. 를 보내 달라. 그리고 이동 방향과 속도도 알려 달라. 수신시 확인 응답할 것.

REQUEST INFO AND LOCATION OF ALL 9TH TD ASSETS AS WELL AS ALL ASSETS OF THE 79TH TD. THESE UNITS APPEAR TO HAVE FLOWN FORWARD OF LAST POSITION.
9 탱크 사단 과 79 탱크 사단에 대한 정보와 그 위치를 알고 싶다. 이들 부대들은 지난번 보다 더 전방으로 이동한 것 같다.

STILL WAITING ON LOCATIONS OF FRIENDLY UNITS.
계속 Friendly Unit의 위치를 수반 중.

NEED EXACT LOCATION AND ATTITUDES OF TAB E/161 RADARS.
표적 취득 포대 E/161 레이다의 정확한 위치와 Attitudes를 알고 싶다.

NEED CURRENT LOCATION OF TP 25.
TP 25의 현재 위치를 알고 싶다.

WHAT IS STATUS OF PREVIOUS REQUEST FOR CENTER MASS OF FA BNS AND HQ CP OF MANEUVER BDE?
전술 여단의 본부 지휘 초소와 야전 포병대대의 Center Mass에 대해 지난번에 요청한 것은 어떻게 되었는가?

PLEASE PASS ABOVE MESSAGE ONTO DIVARTY S3. I CAN NOT REACH THEM ON THE LAN.
위의 Message를 사단포병대 S3에 전해 달라. 우리는 지역통신망을 통하여서는 그들과 접촉할 수가 없다.

MESSAGE RECEIVED ON FSB UNIT LOCATIONS.
전위지원대대의 위치에 관한 Message 수신.

CURRENT LOCATION AND STATUS OF 175 TK REG.
175TK 연대의 현재 위치와 상황.

CURRENT STATUS AND LOCATION OF 180 MRR. ANY INFO ON ENEMY ACTIVITY IN THE VICINITY OF NB4524.
180 자동화기 연대의 현재 위치와 상황. NB 4524 근방의 적군 행동에 관한 정보 요망.

REQUEST CURRENT STATUS AND LOCATION FOR 12 MRR, 120 GMRD.
12 자동화기 연대, 120 자동화기 방위 사단의 현재 위치와 상태를 알고 싶다.

ACKNOWLEDGEMENT NEEDED.

수신 확인 요망.

REQUEST ASPS TO ADD NAI'S FOR THE 149BDE TO GIVE THEM MORE INFORMATION.
더 많은 정보 제공을 하기 위해 149여단에 NAI's를 보강할 것을 군수품 지원대에 요청한다.

REQUEST LOCATIONS FOR THE FOLLOWING UNITS:
다음 부대의 현 위치를 알고 싶다:

I HAVE NOT RECEIVED ENEMY POSITIONS OR ELEMENTS. CAN YOU SUPPLY THAT INFO SO I CAN PLOT ON GRAPHIC?
적군의 위치나 요소에 관한 연락을 아직 받지 못했음. 내가 Graphic을 할 수 있도록 해당 정보를 보내줄 수 있는가?

PLEASE FORWARD THE MESSAGE ABOUT THE MOVEMENT OF THE 79TH TANK DIV.
79탱크 사단의 이동 상황에 대한 메시지를 전송하라.

NEED LOCATION OF EPW COLLECTION POINT FOR BRIGADE.
여단의 적군 포로 수용 지점의 위치를 알고 싶다.

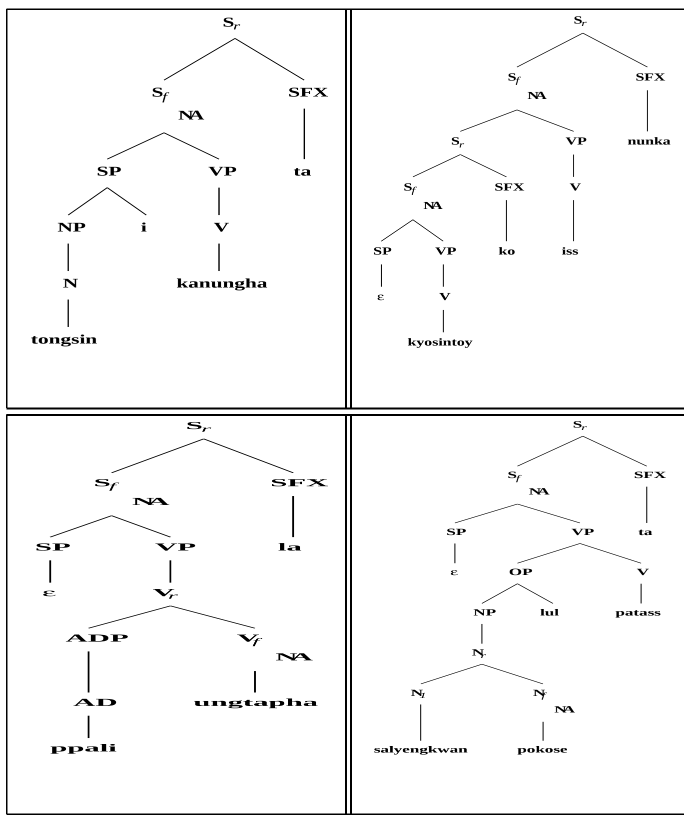
**REQUEST LOC OF 79 TD HQ, ALL REGT. HQS, AND ALL RECON UNITS, AND UNIT STRENGTHS.

**79탱크 사단 본부, 모든 연대 본부, 그리고 모든 수색 부대의 위치와 병력을 알고싶다[6].

[6]The structure for 그리고 is not yet clear as it can have so many structures.

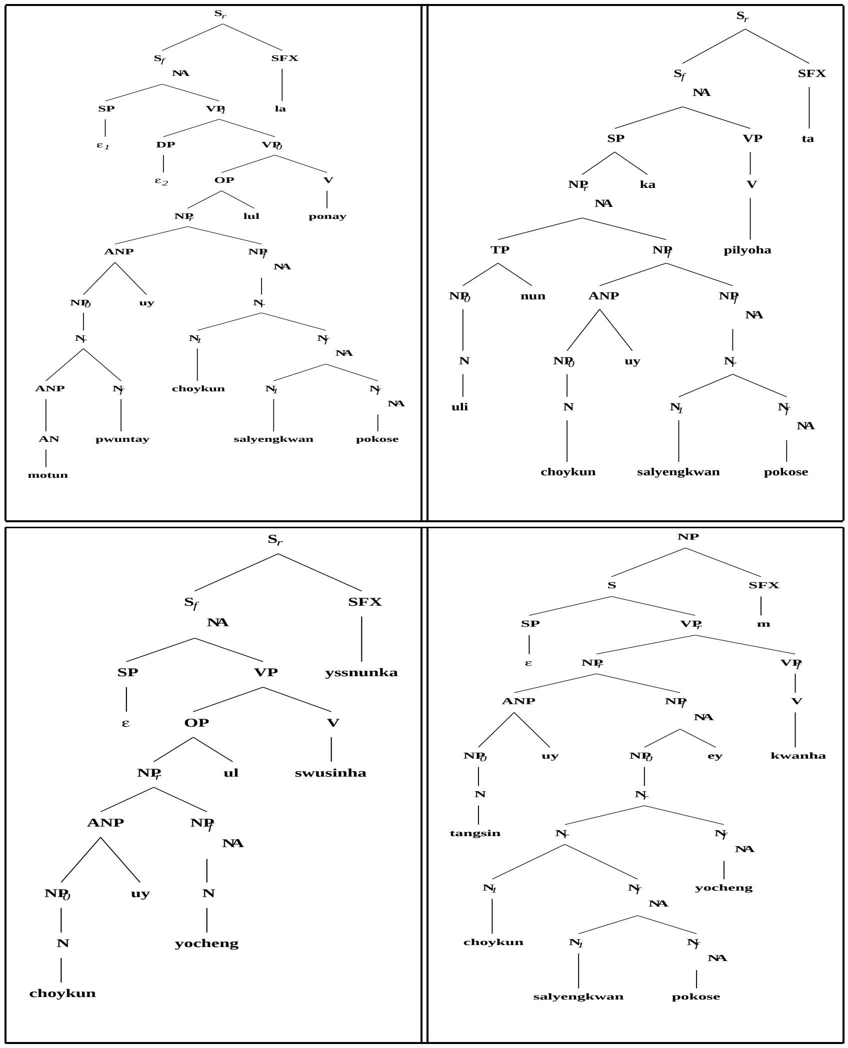
Appendix B

Derived Trees for the Military Messages



[1]

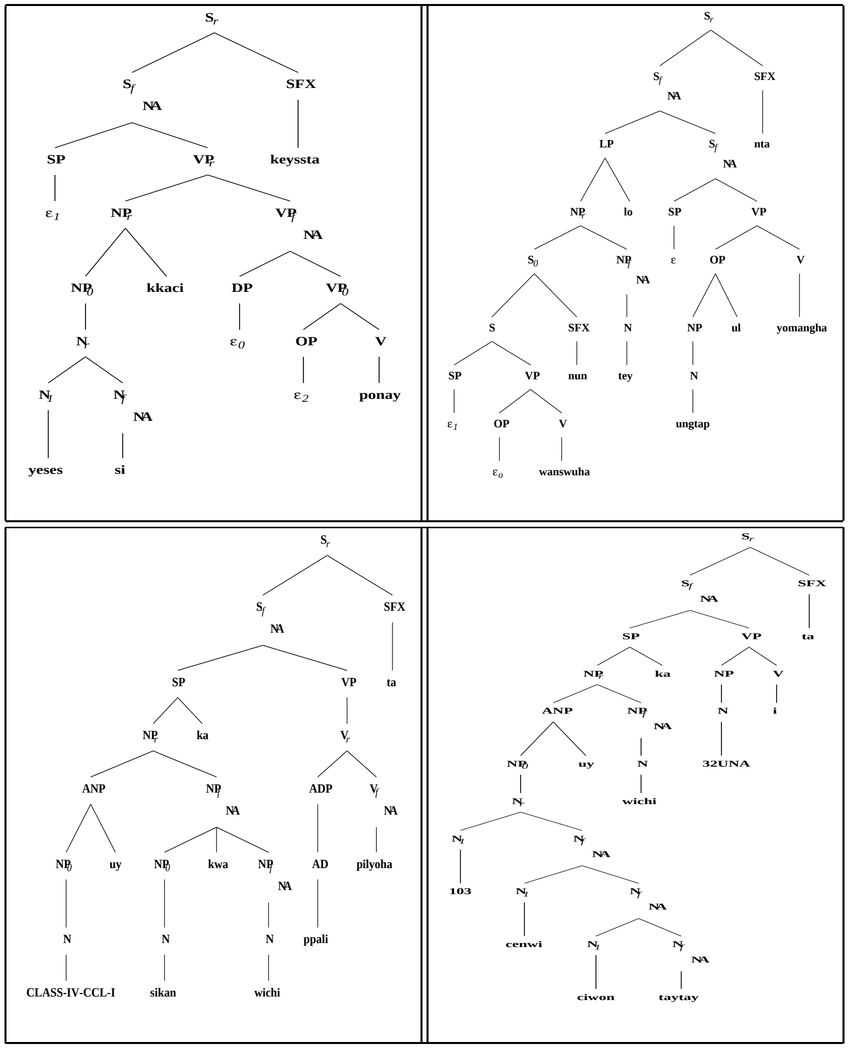
(1)통신이	가능하다.	(2)교신되고	있는가?	
COMMO-NOM	possible-DECL.	Communication-NOM	become ita-PROG	
COMMO	IS UP.	ARE YOU BACK	ON MY LINE?	
(3)빨리	응답하라.	(4)사령관	보고서를	받았다.
ASAP	RESPOND-IMP.	Commander	report-ACC	receive-PAST
RESPOND	ASAP.	WE RECEIVED THE	COMMANDER'S	REPORT.



2

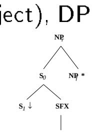
[2]§ (1) 모든	분대의	최근	사령관	보고서를	보내라.
All	unit-POSS	curent	commander	report-ACC	send-IMP
PLEASE	SEND	A CURRENT	CMDRS	REPORT	ON ALL UNITS.
(2) 우리는	최근의	사령관	보고서를	받았다.	
We-TOP	current-POSS	CMDRS	report-ACC	receive-PAST	
WE	RECEIVED	A CURRENT	CMDRS REPORT.		
(3) 최근의	요청을	수신했는가?			
Recent-POSS	request-ACC	receive-PAST-INTR?			
DID YOU	RECEIVE	OUR LAST REQUEST?			
(4) 당신의	최근	사령관	보고서 요청에	관함.	
You-POSS	current	CMDRS	REPORT-LOC	refer-NomSFX	
REFERENCE	TO YOUR	REQUEST	FOR CURRENT	CMDRS REPORT.	

[2]†In (1), *ponay* verb structure has three argument nodes: SP↓, DP↓, and OP↓. SP and DP were substituted with ϵ as they were dropped in the sentence. In (3), subject for the verb *swusinha* is missing. *Choykun* (NP) and *uy* (Adnominal Particle) together are the constituents for ANP node or Adnominal Phrase node. In (4), the whole sentence is a NP, not an S. Suffix *m* here is used for making whole clause into a noun.

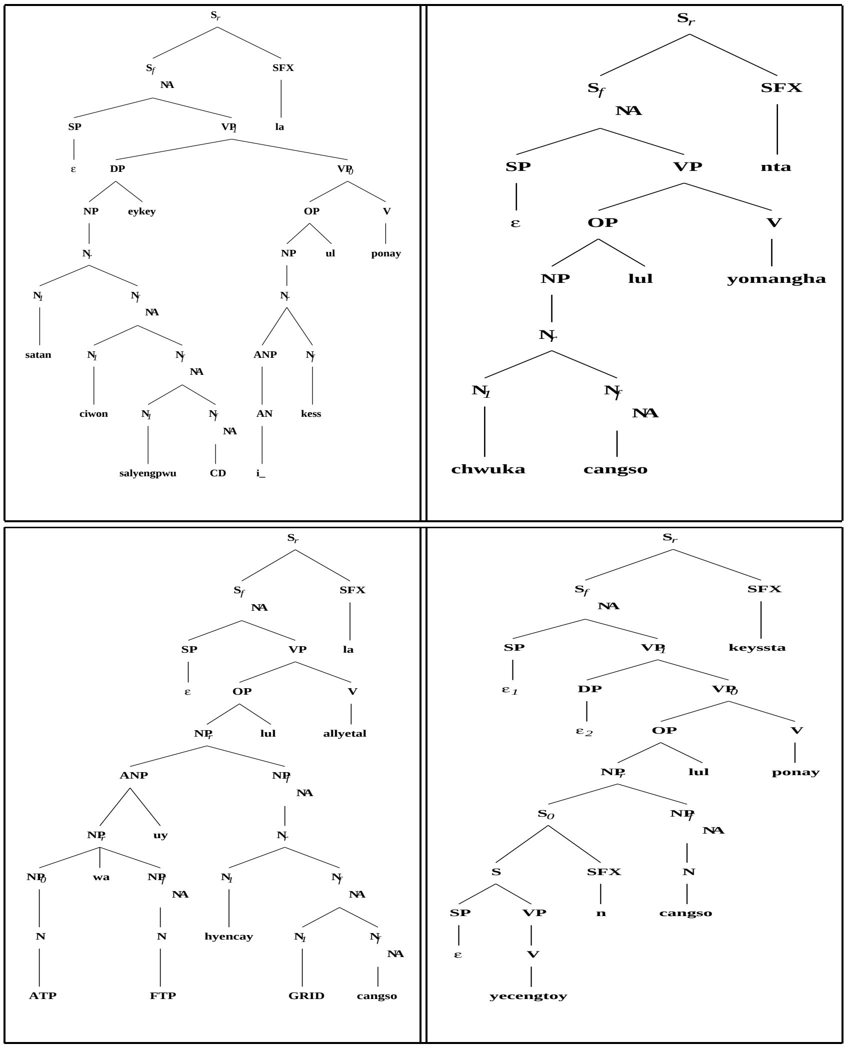


3

[3] § (1) 여섯시 까지 보내겠다.
Six-TIME by-PTC send-WILL-DECL.
WILL SEND BY 0600.
(2) 완수하는 데로 응답을 요망한다.
Fill-ADN DNP-LOC reply-ACC request-DECL.
REQUEST REPLY WHEN THIS IS FILLED.
(3) CLASS IV CCL I의 시간과 위치가 빨리 필요하다.
CLASS IV CCL I-POSS time-AND location-NOM ASAP need-DECL.
NEED TIME AND LOCATION FOR CLASS IV CCL I ASAP.
(4) 103 전위 지원 대대의 위치가 32UNA이다.
103rd FST-POSS location-NOM 32UNA-be-DECL.
LOCATION OF 103RD FSB IS 32UNA.

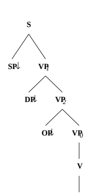
[3] †In (1), three arguments for the verb *ponay* are missing: SP (subject), DP (dative), and OP (object). *yeses si kkaci* is adjoined on VP node, as it is an adverbial phrase. In (2), 

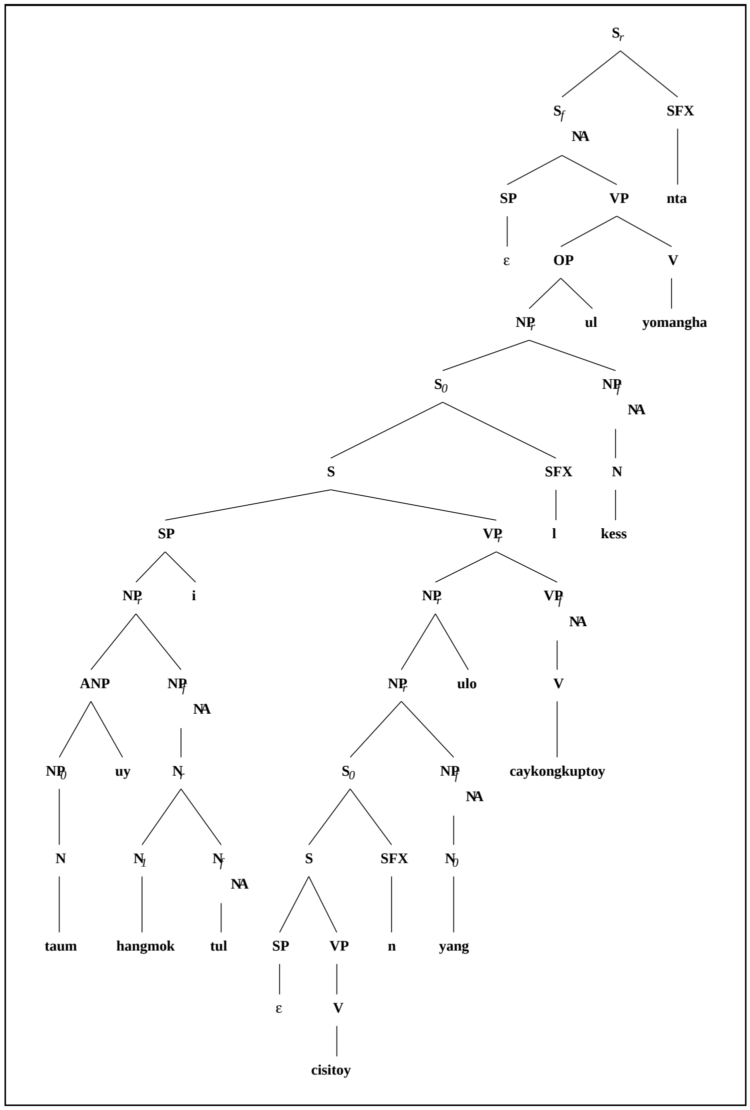
structure was used for *nun*, and this structure is adjoined to modify the dependent noun *tey*.



4

[4]§ (1) 사단 지원 사령부에게 이것을 보내라.
DISCOM CD-DAT THIS-ACC send-IMP.
SEND THIS TO DISCOM CD.
(2) 추가 장소를 요망한다.
Additional location-ACC request-DECL
REQUEST ADDITIONAL LOCATION.
(3) ATP 와 FTP 의 현재 GRID 장소를 알려달라.
ATP AND FTP-POSS current GRID location-ACC request-IMP.
REQUEST CURRENT GRID LOC OF ATP'S AND FTP'S.
(4) 예정된 장소를 보내겠다.
Propose-SFX location-ACC send-IMP
PROPOSED LOC TO FOLLOW.

[4]†In (1), the  structure is used for the verb *ponay*. In (4), 예정된 장소 means information regarding



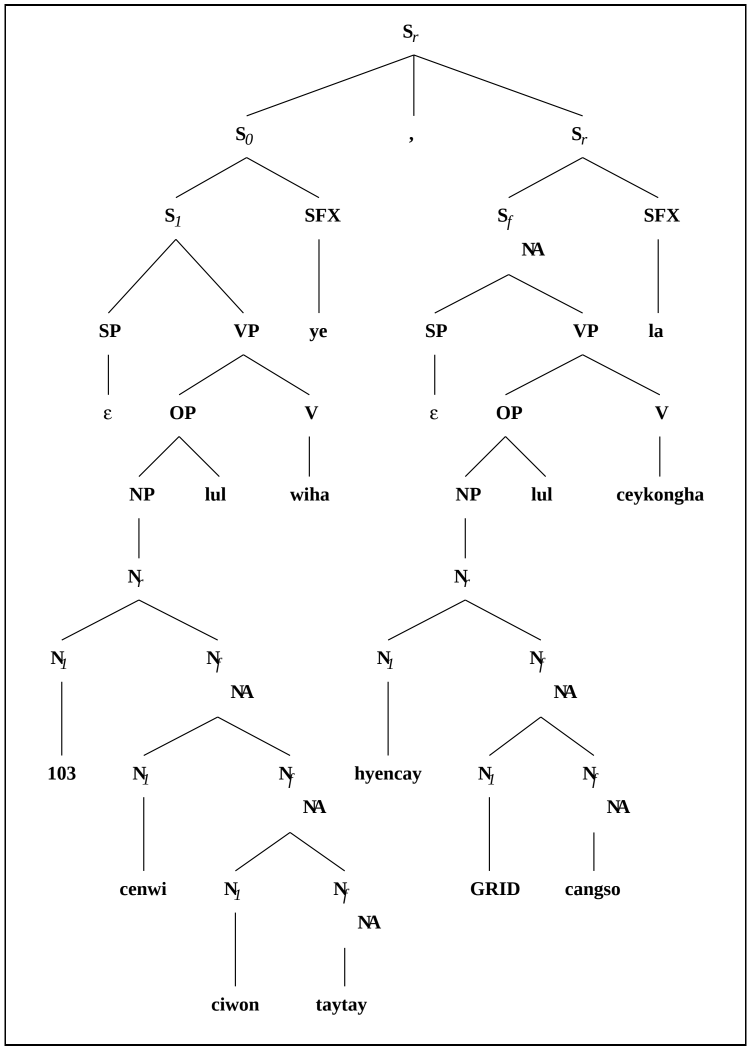
5

---

the proposed location.

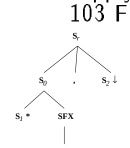
[5] §

다음의	항목들이	지시된	양으로	재공급될	것을	요망한다:
Following-POSS	items-NOM	direct-SFX	quantity-LOC	resupply	DNP-ACC	request-DECL
REQUEST THE	FOLLOWING	ITEMS BE	RESUPPLIED	IN THE	FOLLOWING	QUANTITIES:



[6]

	103 전위 지원대대를	위하여,	현재 GRID 장소를	제공하라.
[6]§	103 FSB-ACC	help-SFX,	current GRID Location-ACC	supply-IMP.
	YOU SUPPLY	CURRENT GRID	LOCATION FOR	103 FSB.

[6]† "For 103 FSB" was translated into a clause in Korean. As a result,  structure was used for handling the conjunctive sentence.

[6]‡ All the trees are derived using XTAG 1.0 system.

Appendix C

Some Possible Scrambled Sentences and their Derived Trees

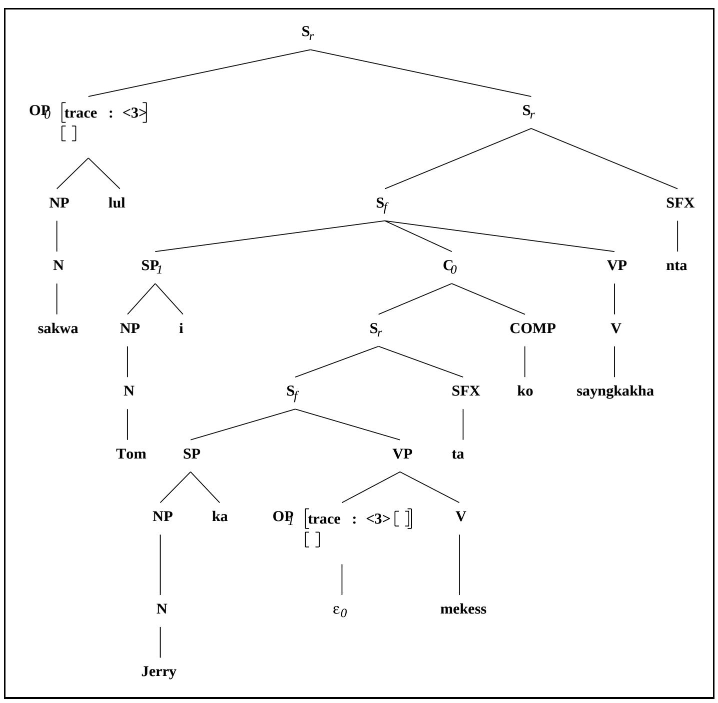


Figure C.1: sakwa-lul Tom-i Jerry-ka mekessta-ko sayngkakhanta

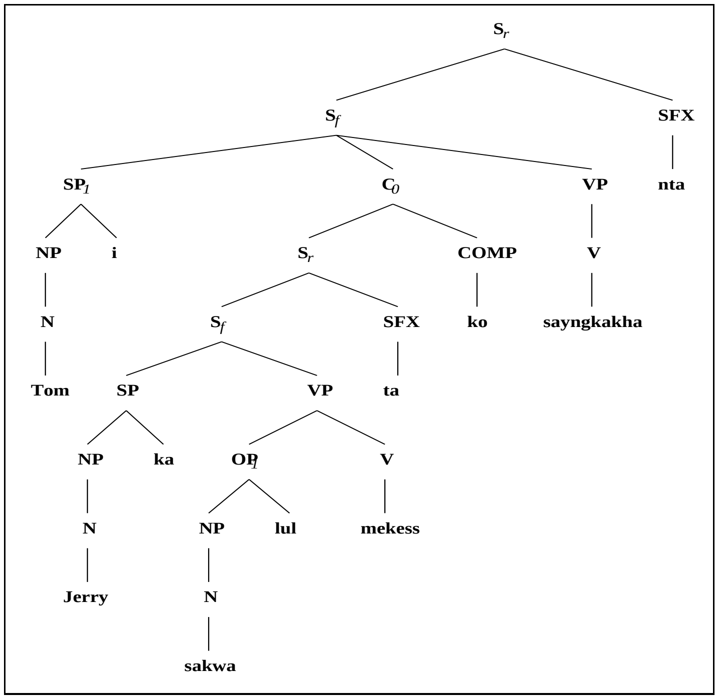


Figure C.2: Tom-i Jerry-ka sakwa-lul mekessta-ko sayngkakhanta

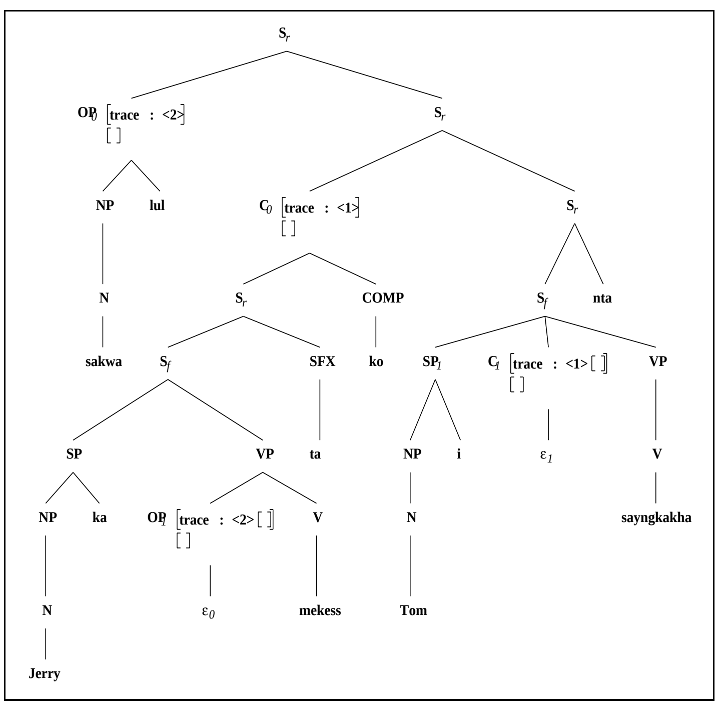


Figure C.3: sakwa-lul Jerry-ka mekessta-ko Tom-i sayngkakhanta

Appendix D

Falsely Indexed Derived Tree by Method $\mathcal{B}$

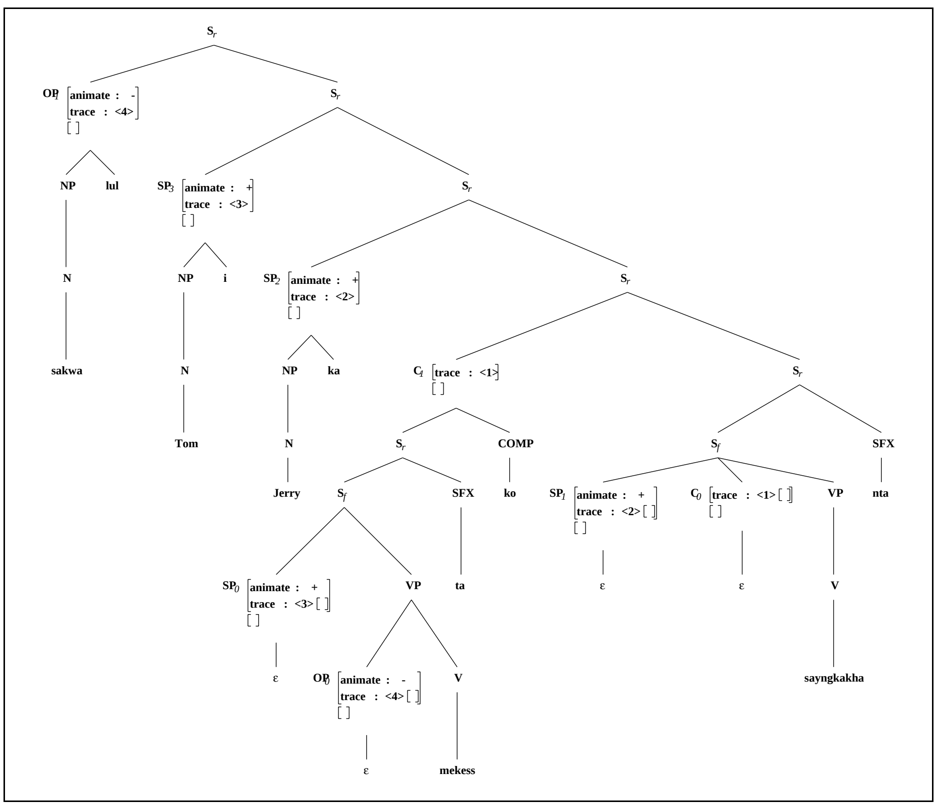


Figure D.1: Falsely Indexed Derived Tree by Method $\mathcal{B}$

Appendix E

The Code List For the Recovery of Empty Arguments

```
:- op(200, xfy, ^).
:- op(300, xfx, :).
:- op(350, yfx, &&).
:- op(400, yfx, \).
:- op(400, yfx, /).
:- op(500, xfy, &).
:- op(510, xfy, =>).

member(X,[X|_]).
member(X,[_|Y]) :-
   member(X,Y).

subset([A|X],Y) :-
   member(A,Y), subset(X,Y).
subset([],Y).

disjoint(X,Y) :-
   not( member(Z,X), member(Z,Y) ).

% No duplicate element for Intersection
intersection([],X,[]).
intersection([X|R],Y,[X|Z]) :-
  member(X,Y),
  !,
  intersection(R,Y,Z).
intersection([X|R],Y,Z) :-
   intersection(R,Y,Z).

fmember(X,[X|_]) :-
   !, fail.
fmember(X^F1,[X^F2|_]) :-
   atom(X).
fmember(X,[_|Y]) :-
   var(Y), !, fail.
fmember(X,[_|Y]) :-
   fmember(X,Y).
```

[0]‡ This code was presented as part of the course work for CIS 630.

```
funion(X,Y,Y) :-
   var(X), !, X = Y.
funion(X,Y,X) :-
   var(Y), !, X = Y.
funion([X|R],Y,Z) :-
   fmember(X,Y), !, fail.
funion([X|R],Y,Z) :-
   member(X,Y), !, funion(R,Y,Z).
funion([X|R],Y,[X|Z]) :-
   funion(R,Y,Z).

union([],X,X).
union([X|R],Y,Z) :-
   member(X,Y), !, union(R,Y,Z).
union([X|R],Y,[X|Z]) :-
   union(R,Y,Z).

% Unification

unify(X:S&&F1,X:S&&F2) :-
   atom(X),
   funion(F1,F2,F3).
unify(X1/Y1,X2/Y2) :-
   unify(X1,X2),
   unify(Y1,Y2).
unify(X1\Y1,X2\Y2) :-
   unify(X1,X2),
   unify(Y1,Y2).

isuni(X,Y,_) :-
   (var(X); var(Y)), !.
isuni([X|R],Y,_) :-
   fmember(X,Y), !, fail.
isuni([X|R],Y,Z) :-
   member(X,Y), !, isuni(R,Y,Z).
isuni([X|R],Y,[X|Z]) :-
   isuni(R,Y,Z).

% Category (category:Semantic&&Feature)

category(nun, s:S&&SF / s:S&&SF \np:N&&NF).
category(un,  s:S&&SF / s:S&&SF \np:N&&NF).

category(na, np:i&&[anim^p, cons^m, food^m | NF]).
category(tom, np:tom&&[anim^p, cons^m, food^m | NF]).
category(ne, np:you&&[anim^p, cons^m, food^m | NF]).
category(ku, np:he&&[anim^p, cons^m, food^m | NF]).
category(salam, np:X^human(X)
                &&[anim^p, cons^m, food^m | NF]).
category(mwuncey, np:X^problem(X)
                &&[anim^m,food^m, cons^m | NF]).

category(i, np:this(X)&&[dist^p | NF]
           /np:X&&[dist^p | NF]).
category(i, np:X^(S1&this(X))&&[dist^p | NF]
           /np:X^S1&&[dist^p | NF]).
category(gin, np:long(X)&&[dist^p | NF]
             /np:X&&[shape^p | NF]).
category(gin, np:X^(S1&long(X))&&[shape^p | NF]
             /np:X^S1&&[shape^p | NF]).

category(ka, s:S&&SF
           / (s:S&&SF\ npn:s(N)&&NF)
           \ np:N&&NF). % >T
category( i, s:S&&SF
```

```
            / (s:S&&SF\ npn:s(N)&&NF)
            \ np:N&&NF). % >T
category(lul, (s:S&&SF\ npn:N&&NF)
             / (s:S&&SF\ npn:N&&NF
               \ npa:o(A)&&[cons^m|AF])
             \ np:A&&[cons^m|AF]).   % >T
category(lul, (s:S&&SF\ npn:N&&NF\ npd:D&&DF)
             / (s:S&&SF\ npn:N&&NF\ npd:D&&DF
               \ npa:o(A)&&[cons^m|AF])
             \ np:A&&[cons^m|AF]).   % >T
category( ul, (s:S&&SF\ npn:N&&NF)
             / (s:S&&SF\ npn:N&&NF\ npa:o(A)&&[cons^p|AF])
             \ np:A&&[cons^p|AF]).   % >T
category( ul, (s:S&&SF\ npn:N&&NF\ npd:D&&DF)
             / (s:S&&SF\ npn:N&&NF\ npd:D&&DF
               \ npa:o(A)&&[cons^p|AF])
             \ np:A&&[cons^p|AF]).   % >T
category(eykey, (s:S&&SF \npn:N&&NF)
             / (s:S&&SF\npn:N&&NF\npd:a(D)&&[anim^p|DF])
             \ np:D&&[anim^p|DF]).   % >T

category(ka, s:S&&SF
           \ (s:S&&SF\ npn: s(N)&&NF) \ np:N&&NF). % <Tx
category( i, s:S&&SF
           \ (s:S&&SF\ npn: s(N)&&NF) \ np:N&&NF). % <Tx
category(lul, (s:S&&SF\ npn:N&&NF)
             \ (s:S&&SF\ npn:N&&NF
                \ npa:o(A)&&[cons^m|AF])
             \ np:A&&[cons^m|AF]).   % <Tx
category(lul, (s:S&&SF\ npn:N&&NF\ npd:D&&DF)
             \ (s:S&&SF\ npn:N&&NF\ npd:D&&DF
                \ npa:o(A)&&[cons^m|AF])
             \ np:A&&[cons^m|AF]).   % <Tx
category( ul, (s:S&&SF\ npn:N&&NF)
             \ (s:S&&SF\ npn:N&&NF
                \ npa:o(A)&&[cons^p|AF])
             \ np:A&&[cons^p|AF]).   % <Tx
category( ul, (s:S&&SF\ npn:N&&NF\ npd:D&&DF)
             \ (s:S&&SF\ npn:N&&NF\ npd:D&&DF
                \ npa:o(A)&&[cons^p|AF])
             \ np:A&&[cons^p|AF]).   % <Tx
category(eykey, (s:S&&SF\npn:N&&NF)
             \ (s:S&&SF\npn:N&&NF\npd:a(D)&&[anim^p|DF])
             \ np:D&&[anim^p|DF]).   % <Tx
category(ko, (s:S&&SF) \ s:S&&SF).

category(cata, s:sleep(s(e))&&[act^p|SF]).
category(cata, s:sleep(X)&&[act^p|SF]
              \ npn:X&&[anim^p|NF]).
category(megta, s:eat(s(e),o(e))&&[act^p|SF]).
category(megta, s:eat(X,o(e))&&[act^p|SF]
              \ npn:X&&[anim^p|NF]).
category(megta, s:eat(s(e),Y)&&[act^p|SF]
              \ npa:Y&&[food^p|AF]).
category(megta, s:eat(X,Y)&&[act^p|SF]
               \ npn:X&&[anim^p|NF]
               \ npa:Y&&[food^p|AF]).
category(pwulswuepta, s:notsolve(s(e),o(e))&&[act^p|SF]).
category(pwulswuepta, s:notsolve(X,o(e))&&[act^p|SF]
                     \ npn:X&&[anim^p|NF]).
category(pwulswuepta, s:notsolve(s(e),Y)&&[act^p|SF]
```

```
                    \ npa:Y&&[anim^m|AF]).
category(pwulswuepta, s:notsolve(X,Y)&&[act^p|SF]
                    \ npn:X&&[anim^p|NF]
                    \ npa:Y&&[anim^m|AF]).
category(malhata, s:say(X,Y)&&[act^p|SF]
                \ npn:X&&[anim^p|NF]
                \ npd:Y&&[anim^p|DF]).
category(sayngkakhanta, s:think(s(e),s:S&&SF)&&[act^m|SF]
                      \ s:S&&SF).
category(sayngkakhanta, s:think(X,s:S&&SF)&&[act^m|SF]
                      \ npn:X&&[anim^p|NF]
                      \ s:S&&SF).
category(cuta, s:give(s(e),a(e),o(e))&&[act^p|SF]).
category(cuta, s:give(s(e),a(e),Z)&&[act^p|SF]
             \ npa:Z&&[anim^m|AF]).
category(cuta, s:give(s(e),Y,o(e))&&[act^p|SF]
             \ npd:Y&&[anim^p|DF]).
category(cuta, s:give(X,a(e),o(e))&&[act^p|SF]
             \ npn:X&&[anim^p|NF]).
category(cuta, s:give(s(e),Y,Z)&&[act^p|SF]
             \ npd:Y&&[anim^p|DF]
             \ npa:Z&&[anim^m|AF]).
category(cuta, s:give(X,a(e),Z)&&[act^p|SF]
             \ npn:X&&[anim^p|NF]
             \ npa:Z&&[anim^m|AF]).
category(cuta, s:give(X,Y,o(e))&&[act^p|SF]
             \ npn:X&&[anim^p|NF]
             \ npd:Y&&[anim^p|DF]).
category(cuta, s:give(X,Y,Z)&&[act^p|SF]
             \ npn:X&&[anim^p|NF]
             \ npd:Y&&[anim^p|DF]
             \ npa:Z&&[anim^m|AF]).

% reduce rule necessary for Korean.
% First rule is for storing topic on the local stack.

reduce(np:N1&&NF1,s:S&&SF / s:S&&SF
      \np:N2&&NF, s:S&&SF / s:S&&SF) :-
   unify(np:N1&&NF1,np:N2&&NF2),
   lstack(LSTK),
   retract(lstack(_)),
   assertz(lstack([np:N1&&NF1|LSTK])).

reduce(X/Y1, Y2, X) :-
   unify(Y1,Y2).              % >
reduce(Y1, X\Y2, X) :-
   unify(Y1,Y2).              % <
reduce(X/Y1, Y2/Z, X/Z) :-
   unify(Y1,Y2).              % >B
reduce(X/Y1, Y2\Z, X\Z) :-
   unify(Y1,Y2).              % >Bx
reduce(Y1/Z, X\Y2, X/Z) :-
   unify(Y1,Y2).              % <Bx
reduce(Y1\Z, X\Y2, X\Z) :-
   unify(Y1,Y2).              % <B

% Parse simulates reduce-first SR parser with backtracking:
% ex) parse([],[na,nun,i,mwunce,nun,pwulswuepta,ko,sayngkakhanta],S).
% This version includes  type raising:
% It is handling recovering empty arguments

parse(Buffer, Result) :-
   parse([], Buffer, Result).
```

```
parse([Cat2, Cat1|Stack], Buffer, Result) :-
  reduce(Cat1, Cat2, Cat3),
  parse([Cat3|Stack], Buffer, Result).
parse([Result], [], Result) :-
   context(Ctext),
   retract(context(_)),
   assertz(context([Result|Ctext])).
parse(Stack, [Word|Buffer], Result) :-
   category(Word, Cat),
   parse([Cat|Stack], Buffer, Result).

% check if the appropriate argument is on the stack

check(NF,Y) :-
   lstack(LS),
   check(LS,NF,Y).

check([SP:X&&YF],NF,Y) :-
   isuni(YF,NF,_), !, X = Y.
check([SP:X&&YF|REM],NF,Y) :-
   isuni(YF,NF,_), !, X = Y.
check([SP:X&&YF|REM],NF,Y) :-
   not isuni(YF,NF,_), !,
   check(REM,NF,Y).

% recovering empty arguments

recover :-
   context([C|REM]),
   C = s:S&&SF,
   S =.. L,
   recover(L,R),
   retract(context(_)),
   assertz(context([(s:R&&SF)|REM])).

% 1. f(s(e),o(e)) type.
recover([F,s(e),o(e)],R) :-
   V =.. [F,_,_],
   category(_,s:V&&VF \ npn:N&&NF \ _),
   check(NF,Y),
   recover([F,s(Y),o(e)],R).

% 2. f(s(e)) type.
recover([F,s(e)],R) :-
   V =.. [F,_],
   category(_,s:V&&VF \ npn:N&&NF),
   check(NF,Y),
   R =.. [F,s(Y)].

% 3. f(s(e),o(OP)) type.
recover([F,s(e),o(OP)],R) :-
   V =.. [F,_,_],
   category(_,s:V&&VF \ npn:N&&NF \ _),
   check(NF,Y),
   R =.. [F,s(Y),o(OP)].

% 4. f(s(SP),o(e)) type.
recover([F,s(SP),o(e)],R) :-
   V =.. [F,_,_],
   category(_,s:V&&VF \ _ \ npa:A&&AF),
   check(AF,Y),
   R =.. [F,s(SP),o(Y)].

% 5. f(s(e), s:S&&SF) type.
recover([F,s(e),s:S1&&S1F],R) :-
   V =.. [F,_,_],
   category(_,s:V&&VF \ npn:N&&NF \ _),
```

```
   check(NF,Y),
   S1 =.. SS,
   recover(SS,RR),
   R =.. [F,s(Y),s:RR&&S1F].

% 6. f(s(e),a(AP),o(OP)) type.
recover([F,s(e),a(AP),o(OP)],R) :-
   V =.. [F,_,_,_],
   category(_,s:V&&VF \ npn:N&&NF \ _ \ _),
   check(NF,Y),
   R =.. [F,s(Y),a(AP),o(OP)].

% 7. f(s(SP),a(e),o(OP)) type.
recover([F,s(SP),a(e),o(OP)],R) :-
   V =.. [F,_,_,_],
   category(_,s:V&&VF \ _ \ npd:D&&DF \ _),
   check(DF,Y),
   R =.. [F,s(SP),a(Y),o(OP)].

% 8. f(s(SP),a(AP),o(e)) type.
recover([F,s(SP),a(AP),o(e)],R) :-
   V =.. [F,_,_,_],
   category(_,s:V&&VF \ _ \ _ \npa:A&&AF),
   check(AF,Y),
   R =.. [F,s(SP),a(DP),o(Y)].

% 9. f(s(e),a(e),o(OP)) type.
recover([F,s(e),a(e),o(OP)],R) :-
   V =.. [F,_,_,_],
   category(_,s:V&&VF \ npn:N&&NF \ _ \ _),
   check(NF,Y),
   recover([F,s(Y),a(e),o(OP)],R).

% 10. f(s(e),a(AP),o(e)) type.
recover([F,s(e),a(AP),o(e)],R) :-
   V =.. [F,_,_,_],
   category(_,s:V&&VF \ npn:N&&NF \ _ \ _),
   check(NF,Y),
   recover([F,s(Y),a(AP),o(e)],R).

% 11. f(s(SP),a(e),o(e)) type.
recover([F,s(SP),a(e),o(e)],R) :-
   V =.. [F,_,_,_],
   category(_,s:V&&VF \ _ \ npd:D&&DF \ _),
   check(DF,Y),
   recover([F,s(SP),a(Y),o(e)],R).

% 12. f(s(e),a(e),o(e)) type.
recover([F,s(e),a(e),o(e)],R) :-
   V =.. [F,_,_,_],
   category(_,s:V&&VF \ npn:N&&NF \ _ \ _),
   check(NF,Y),
   recover([F,s(Y),a(e),o(e)],R).

% Buffer
context([]).
lstack([]).

P&Q :- P, Q.
P=>Q :- \+ P; Q.
exists(X, P&Q) :- P, Q.
forall(X, P=>Q) :- \+ exists(X,P& (\+ Q)).
```

```
Script started on Mon May  2 13:40:04 1994

> cprolog

C-Prolog version 1.5

| ?- ['ccg.pl'].

ccg.pl consulted 20112 bytes 0.133333 sec.

| ?- parse([na,nun,i,mwuncey,nun,pwulswuepta,
            ko,sayngkakhanta],S).
 %% (I) think (I) cannot solve this problem

S = s:think(s(e),s:notsolve(s(e),o(e))&&[act^p|_225])
                &&[act^m,act^p|_225]


yes

| ?- listing(context).

context([s:think(s(e),s:notsolve(s(e),o(e))&&[act^p|_10])
                     &&[act^m,act^p|_10]]).

yes

| ?- listing(lstack).

lstack([np:this(_10^problem(_10))
           &&[dist^p,anim^m,food^m,cons^m,dist^p|_11],
        np:na
           &&[anim^p,cons^m,food^m|_12]]).

yes

| ?- recover.

yes

| ?- listing(context).

context([s:think(s(na),s:notsolve(s(na),
                      o(this(_10^problem(_10))))
                        &&[act^p|_11])
                &&[act^m,act^p|_11]]).

yes

| ?- halt.
```

```
[ Prolog execution halted ]

> exit

script done on Mon May  2 13:41:18 1994
```

Bibliography

[1] Anne Abeillé. French pronomial clitics and synchronous tags. In *TAG+ Workshop - Abstracts*, University of Pennsylvania, 1992. Institute for Research in Cognitive Science.

[2] Anne Abeillé, Yves Schabes, and Aravind K. Joshi. Using Lexicalized Tags for Machine Translation. In *Proceedings of the International Conference on Computational Linguistics (COLING '90)*, Helsinki, Finland, 1990.

[3] Kazimierz Ajdukiewicz. *Die syntaktische Konnexitat*. Oxford University Press, 1935.

[4] Tilman Becker. *HyTAG: A New Type of Tree Adjoining Grammars for Hybrid Syntactic Representation of Free Word Order Languages*. PhD thesis, University of Saarbrücken, January 1994.

[5] Tilman Becker, Aravind Joshi, and Owen Rambow. Long Distance Scrambling and Tree Adjoining Grammars. In *Proceedings of the 5th Conference of the European Chapter*. ACL, 1991.

[6] Tilman Becker and Owen Rambow. Long-Distance Scrambling in German. Technical report, University of Pennsylvania, 1990.

[7] Tilman Becker, Owen Rambow, and Michael Niv. The derivational generative power, or, scrambling is beyond LCFRS. Technical report, University of Pennsylvania, 1992. A version of this paper was presented at MOL3, Austin, Texas, November 1992.

[8] Owen Bratt. Clause Structure and Case Marking in Korean Causatives. *Harvard Studies in Korean Linguistics*, V:241 – 251, 1993.

[9] Jae Ohk Cho. The interaction of Syntax and Morphology in Korean. *Harvard Studies in Korean Linguistics*, II:27 – 39, 1987.

[10] Seungja Choi. Explanations of Negations in Korean. *Harvard Studies in Korean Linguistics*, I:124 – 134, 1985.

[11] Noam Chomsky. *Lectures on Government and Binding*. Foris, 1981.

[12] Deborah A. Dahl and Catherine N. Ball. Reference resolution in PUNDIT. In P. Saint-Dizier and S. Szpakowicz, editors, *Logic and Logic Grammars for Language Processing*. Ellis Horwood, Chichester, England, 1990.

[13] Bonnie Jean Dorr. *Machine Translation: A View from the Lexicon*. MIT Press, Cambridge, Mass, 1993.

[14] D. Egedi, M. Palmer, H.S. Park, and A. Joshi. Korean To English Translation Using Synchronous TAGs. In *the First Conference of the Association for Machine Translation in the Americas*, 1994.

[15] Sergei Nirenburg et al. *Machine Translation: A Knowledge-Based Approach*. Morgan Kaufmann, 1992.

[16] G. Gazdar, E. Klein, G. Pullum, and I. Sag. *Generalized Phrase Structure Grammar*. Harvard University Press, Cambridge, Massachusetts, 1985.

[17] Barbara J. Grosz. The Representation and Use of Focus in Dialogue Understanding. Technical report, SRI International, 1977.

[18] Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. Centering: A Framework for Modelling the Local Coherence of Discourse. *To appear in Computational Linguistics*, 1994.

[19] Barbara J. Grosz and Candace L. Sidner. Attention, intentions and the structure of discourse. *Computational Linguistics*, 12, 1986.

[20] B.J. Grosz, A.K. Joshi, and S. Weinstein. Providing A Unified Account Of Definite Noun Phrases in Discourse. *Proceedings of the 21st Annual Meeting of The Association for Computational Linguistics*, 1983.

[21] Suk-In Chang Ho-Bin Ihm, Kyung-Pyo Hong. *Korean Grammar for International Learners*. Yonsei University Press, 1988.

[22] B.A. Hockey and B. Srinivas. Feature-Based TAGs In Place Of Multi-Component Adjunction: Computational Implications. Technical report, University of Pennsylvania, 1994.

[23] Beryl Hoffman. A CCG Approach to Free Word Order Languages. In *the International Workshop on NL Generation*, 1992.

[24] Aravind Joshi and Martha Palmer. Experimenting with Lexical Semantics and Synchronous TAGs. Presentation at Japan-US Workshop on Machine Aided Translation, 1993.

[25] Aravind K. Joshi. Tree Adjoining Grammars: How much context Sensitivity is required to provide a reasonable structural description. In D. Dowty, I. Karttunen, and A. Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press, Cambridge, U.K., 1985.

[26] Aravind K. Joshi, L. Levy, and M. Takahashi. Tree Adjunct Grammars. *Journal of Computer and System Sciences*, 1975.

[27] Aravind K. Joshi and B. Srinivas. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. In *Proceedings of the* 17^{th} *International Conference on Computational Linguistics (COLING '94)*, Kyoto, Japan, August 1994.

[28] Megumi Kameyama. *Zero Anaphora: the Case of Japanese.* PhD thesis, Stanford University, 1985.

[29] Anthony Kroch and Aravind K. Joshi. Analyzing Extraposition in a Tree Adjoining Grammar. In G. Huck and A. Ojeda, editors, *Discontinuous Constituents, Syntax and Semantics*, volume 20. Academic Press, 1987.

[30] S. Kuno. *The Structure of the Japanese Language*. MIT Press, 1973.

[31] Chang-In Lee. Machine Translation of English Passives into Korean. *Harvard Studies in Korean Linguistics*, 1987.

[32] Ik-Hwan Lee. Toward a Proper Treatment of Scrambling in Korean. *Harvard Studies in Korean Linguistics*, pages 190 – 205, 1985.

[33] Young-Suk Lee. A Korean Causative: A TAG Analysis. Technical Report MS-CIS-89-19, University of Pennsylvania, 1989.

[34] Young-Suk Lee. *Scrambling as a Case-Driven Obligatory Movement*. PhD thesis, University of Pennsylvania, 1993.

[35] Young-Suk Lee and Michael Niv. Scrambling and Coordination in Korean. Technical report, University of Pennsylvania, 1989.

[36] Gui-Sun Moon. The Distribution of Pro in Korean and the Theoretical Implications. Technical report, University of Texas at Austin, 1987.

[37] Gui-Sun Moon. *The Syntax of Null Arguments with Special Reference to Korean*. PhD thesis, University of Texas at Austin, 1990.

[38] Sergei Nirenburg and Robert Frederking. Toward Multi-Engine Machine Translation. Unpublished Paper, 1994.

[39] Martha Palmer, Rebecca Passonneau, Carl Weir, and Tom Finin. The KERNEL text understanding system. *Artificial Intelligence*, 63:17–68, 1993.

[40] B.S. Park. On the Multiple Subject Constructions in Korean. *Linguistics*, pages 63 – 76, 1973.

[41] Hyun-Seok Park, Dania Egedi, and Martha Palmer. Recovering Empty Arguments in Korean. *The Joint Conference of the 8th Asian Conference on Language, Information, and Computation and the 2nd Pacific Asia Conference on Formal and Computational Linguistics* , 1994.

[42] Carl Pollard. Generalized Phrase Structure Grammar, Head Grammar, and Natural Language. Ph.D. thesis, Stanford University, 1990.

[43] Owen Rambow. *Formal and Computational Models for Natural Language Syntax*. PhD thesis, University of Pennsylvania, 1994.

[44] Owen Rambow and Young-Suk Lee. Word Order Variation and Tree Adjoining Grammar. Technical report, University of Pennsylvania, 1991.

[45] Mamoru Saito. *Some Asymmetries in Japanese and Their Theoretical Implications*. PhD thesis, MIT, 1985.

[46] Mamoru Saito. Scrambling as Semantically Vacuous A'-Movement. In Mark R. Baltin and Anthony S. Kroch, editors, *Alternative Concepts of Phrase Structure*. The University of Chicago Press, 1989.

[47] Mamoru Saito. Long Distance Scrambling in Japanese, 1990. University of Connecticut.

[48] Yves Schabes. Mathematical and computational aspects of lexicalized grammars. Ph.D. thesis MS-CIS-90-48, LINC LAB179, Computer Science Department, University of Pennsylvania, Philadelphia, PA, 1990.

[49] Yves Schabes, Anne Abeillé, and Aravind K. Joshi. Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars. In *Proceedings of the* 12^{th} *International Conference on Computational Linguistics (COLING'88)*, Budapest, Hungary, August 1988.

[50] Yves Schabes, Patrick Paroubek, and the XTAG Research Group. *XTAG User Manual version 1.0*. University of Pennsylvania, Philadelphia PA 19104-6389, 1994.

[51] Stuart Shieber and Yves Schabes. Synchronous Tree Adjoining Grammars. In *Proceedings of the* 13^{th} *International Conference on Computational Linguistics (COLING'90)*, Helsinki, Finland, 1990.

[52] Stuart Shieber and Yves Schabes. Generation and Synchronous Tree Adjoining Grammars. In *Proceedings of the Fifth International Workshop on Natural Language Generation*, Pittsburgh, Pennsylvania, 1991.

[53] Candace L. Sidner. *Towards a Computational Theory of Definite Anaphora*. PhD thesis, Massachusetts Institute of Technology, Cambridge,MA, 1979.

[54] Mark Steedman. Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory*, 5:403–439, 1987.

[55] K. Vijay-Shanker. *A Study of Tree Adjoining Grammars*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1987.

[56] K. Vijay-Shanker and Aravind K. Joshi. Unification Based Tree Adjoining Grammars. In J. Wedekind, editor, *Unification-based Grammars*. MIT Press, Cambridge, MA, 1991.

[57] K. Vijay-Shanker, D.J. Weir, and A.K. Joshi. Tree Adjoining and Head Wrapping. In *Proceedings of the* 11^{th} *International Conference on Computational Linguistics (COLING '86)*, 1986.

[58] Mariln Walker, Masayo Iida, and Sharon Cote. Japanese Discourse and the Process of Centering. Technical report, University of Pennsylvania, 1992.

[59] David J. Weir. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, University of Pennsylvania, 1988.

[60] Yorick Wilks. Stone soup and the French room. In A. Zompoli, N. Calzolari, and M. Palmer, editors, *Current Issues in Natural Language Processing: In Honour of Don Walker*. Giardini with Kluwer, 1994.

[61] Zhibiao Wu and Martha Palmer. Verb Semantics and Lexical Selection. *32nd Meeting of the Association for Computational Linguistics*, 1994.

[62] In-Seok Yang. *Korean Syntax: Case Markers, Delimiters, Complementations, and Relativization*. PhD thesis, University of Hawaii, 1971.