Reducing Syntactic Ambiguity in the K-TCCG System

Yong-hun Lee
(Chungnam National University · Hannam University)

Lee, Yong-hun. 2011. Reducing Syntactic Ambiguity in the K-TCCG System. *The Linguistic Association of Korea Journal. 19*(2). 79-103. Syntactic ambiguity in parse trees is one of the crucial problems when we computationally implement the syntactic and semantic analysis of natural languages in the Type-inherited Combinatory Categorial Grammar (TCCG). In order to solve this problem, Beavers (2002, 2004) introduced the type Normal Form *nf*, which imposes some constraints on the possible structures of the parse trees. However, Beavers' type hierarchy of *nf* doesn't fit into the K-TCCG (Korean TCCG) system because of head parameters. The goal of this paper is to solve this problem by developing a modified type hierarchy of *nf* so that the K-TCCG system can correctly implement various syntactic phenomena in the Korean language. In this modified type hierarchy of *nf* and the constraints on category combinatorics, the RNR constructions as well as simple sentences can be effectively implemented without any syntactic ambiguity in parse trees.

Key Words: syntactic ambiguity, parse tree, normal form, type hierarchy, K-TCCG

1. Introduction

The efficiency problem which is raised by syntactic ambiguity in parse trees is one of the most important problems in the computational implementation of the syntactic and semantic analyses of natural languages in the TCCG system. The problem is raised in the system since (i) Combinatory Categorial Grammar (CCG; Steedman, 1996, 2000) itself has five different category combinatorics (*functional application, functional composition, type raising, functional substitution,* and *co-ordination*) and (ii) more than one category combinatorics may be applied

when one constituent combines another constituent. The efficiency problem in parse trees may not be a problem in the theoretical analysis of the natural languages. However, it is one of the most fundamental problems in the computational implementation, since syntactic ambiguity in parse trees significantly increases the time and space complexity when we parse the sentences of the specific natural languages.

In order to solve this problem, Beavers (2002, 2004) introduced the type Normal Form *nf* and the type hierarchy for this type. The type *nf* encodes the information on what kind of category combinatorics is available when one constituent combines another constituent, and it imposes some constraints on the possible types of category combinatorics in the next step. By adopting the type *nf* and its type hierarchy, he succeeded to drastically reduce the syntactic ambiguity of English parse trees (in his English TCCG system) within the reasonable time and space.

However, Beavers' type hierarchy of *nf* doesn't fit into the K-TCCG (Korean TCCG) system because head parameters work between two languages. It is well known that English is a head initial language whereas Korean is a head-final language. Since Korean is a head-final language, a predicate (an adjective or a verb) is located at the end of the sentence, and an NP may combine with another NP before it combines with a predicate. It makes the syntactic ambiguity problem even in the simple sentences.

The goal of this paper is to solve this syntactic ambiguity problem in the K-TCCG system. In order to solve this problem, this paper modifies the type hierarchy of *nf* so that the K-TCCG system can properly implement various syntactic phenomena in the Korean language. As you will see, one of the problematic constructions in Korean when we modify the type hierarchy of *nf* is the Right Node Raising (RNR) constructions, where the operations *type raising* and *forward functional application* are used. Accordingly, we have to take the RNR constructions into consideration during the modification of the type hierarchy of *nf*, in addition to other constructions. Therefore, in the modified type hierarchy of *nf*, the RNR constructions as well as simple sentences can be effectively implemented without any syntactic ambiguity in parse trees.

2. Syntactic Ambiguity in the English TCCG System

2.1. Grammar Engineering and the English TCCG System

Since the Linguistic Knowledge Building grammar development system (LKB, Copestake, 2002) was developed by many scholars such as Ann Copestake, John Carroll, Rob Malouf, and Stephen Oepen, there have been many trials to computationally implement Steedman's CCG system using the LKB system. Because the LKB system is framework independent (Copestake, 2002, p. 6) and it is designed as a tool to develop any typed, feature-based, unification grammar regardless of underlying formal grammars (Beavers, 2002, p. 4), it is also possible to implement the CCG in the LKB system.

The first trial implementation of CCG was done by Ann Copestake, and a small part of it was included in the distributed version of the LKB system. Aline Vallavicemcio (2001) further developed the implemented grammar for CCG, and it formed a fundamental basis for the English TCCG system. Beavers (2002, 2004) developed a TCCG system for English fragments, and various syntactic phenomena in English were computationally implemented in the English TCCG system, which included *co-ordination, control, unbounded dependencies, passive, imperatives, bare plurals, mass nouns, dative shift, extraposition, auxiliaries, predicatives, expletives,* and *there-insertion.* As mentioned in Beavers (2002, p. 4), his system was theoretically based on Carpenter (1992), Steedman (1996, 2000), Baldridge (2002), and Hockenmaier et al. (2001). In the actual implementation, however, his system was primarily based on Vallavicemcio (2001).

Beavers' (English) TCCG system is based on the concept of grammar engineering. As mentioned in Bender (2008, p. 21), *grammar engineering* is the practice of building elaborated linguistic models on computer. There are roughly two types of applications of grammar engineering in linguistics. First, it can be used for linguistic hypothesis testing, as Bender (2008) suggested. Implemented grammars enable the linguistic hypotheses to be tested against thousands of real example sentences. Through the implementation of a formal grammar, we linguists can identify and correct the problems in our linguistic hypotheses. Accordingly, we are able to make our hypotheses stronger and more reasonable. Second, implemented grammars can be used in natural language processing. For

example, the implemented grammars in the LinGO Grammar Matrix (Bender et al., 2002; Bender and Flickinger, 2005) are currently used for natural language processing, and the machine translation systems using these grammars have already been developed between English and German (Copestake et al., 1995) and between English and Japanese (Bond et al., 2005).

Then, why is grammar engineering necessary for linguistic analyses. Bender (2008:15) provided the answers to this question as follows: "First, languages are made up of many subsystems with complex interactions. Linguists generally focus on just one subsystem at a time, yet the predictions of any particular analysis cannot be calculated independently of the interacting subsystems. With implemented grammars, the computer can track the effects of all aspects of the implementation while the linguist focuses on developing just one. Second, automated application of grammars to test suites and naturally occurring data allows for much more thorough testing of linguistic analysis — against thousands as opposed to tens of examples and including examples not anticipated by the linguist."

As Bender (2008, p. 22-23) pointed out, the basic requirements for *grammar engineering* are (i) a reasonably stable grammar formalism, (ii) algorithms for parsing (and ideally also generation), (iii) grammar development tools, and (iv) test suite management software. Among them, the third one refer to a suite of grammar visualization and debugging aides, and the last one means softwares which facilitates the batch processing of test examples and comparison of results across different grammar versions.

Beavers' TCCG system fits into these criteria. First, it incorporates a reasonably stable grammar formalism, i.e. Steedman's CCG. Second, it includes algorithms for parsing and generation. Third, it contains grammar development tools, the LKB system. Finally, it has a test suite management software, a batch processing tool PET (Callmeier, 2001) and a test suite management tool [incr tsdb()] (Oepen, 2002). Therefore, we can say that Beavers' TCCG system satisfies all the criteria basic requirements for grammar engineering.

2.2. Normal Form and Syntactic Ambiguity in Beavers' TCCG System

Beavers (2002, 2004) also mentioned the efficiency problem in his English

TCCG system, In order to see what is the problem, let's see the following simple sentence.

(1) Kim eats pizza.

For this simple sentence, five possible analyses in Figure 1 are possible (Beavers, 2002, p. 99).[1)]

(2) Functional Application (Steedman, 1996, p. 13)

a. $X / Y : f \quad Y : a \rightarrow X : f\,a \quad (>)$

b. $Y : a \quad X \backslash Y : f \rightarrow X : f\,a \quad (<)$

(3) Functional Composition (Steedman, 1996, p. 43)

a. $X / Y : f \quad Y / Z : g \rightarrow_{\mathbf{B}} X / Z : \lambda x.f(gx) \quad (> \mathbf{B})$

b. $X / Y : g \quad Y \backslash Z : f \rightarrow_{\mathbf{B}} X \backslash Z : \lambda x.g(fx) \quad (> \mathbf{B})$

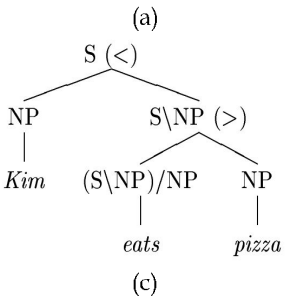
c. $Y \backslash Z : f \quad X \backslash Y : g \rightarrow_{\mathbf{B}} X \backslash Z : \lambda x.g(fx) \quad (< \mathbf{B})$

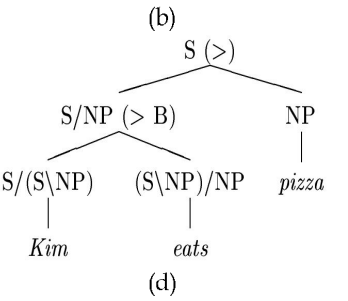
d. $Y / Z : g \quad X \backslash Y : f \rightarrow_{\mathbf{B}} X / Z : \lambda x.f(gx) \quad (< \mathbf{B})$

(4) Type Raising (Steedman, 1996, p. 37)

a. $X : a \rightarrow_{\mathbf{T}} T / (T \backslash X) : \lambda f.fa \quad (> \mathbf{T})$

b. $X : a \rightarrow_{\mathbf{T}} T \backslash (T / X) : \lambda f.fa \quad (< \mathbf{T})$





1) Beavers did not include determiners in the analyses for convenience.

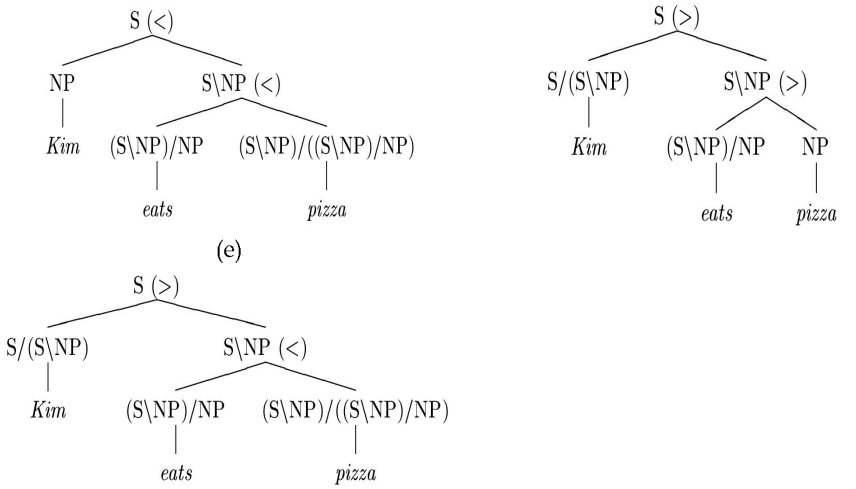


Figure 1. Four Possible CCG Analyses for the English Sentence in (1)

These five analyses are constructed as follows. In the analysis (a), two *functional application*s are applied. After the *forward functional application* in (2a) is applied to combine a transitive *eats* and the object NP *pizza*, the *backward functional application* in (2b) is applied to combine the subject NP *Kim* and the VP *eats pizza*. In the analysis (b), after the subject NP *Kim* is *forward type raised* by (3a), it combines with a transitive *eats* by a *forward functional composition* in (3a). Then, this constituent combines with the object NP *pizza* by the *forward functional application* in (2a). In the analysis (d), first of all, a transitive *eats* and the object NP *pizza* are combined by the *forward functional application* in (2a). Then, the subject NP *Kim* is *forward type raised* by (3a). Then, it combines with a VP *eats pizza* once again by a *forward functional application* in (2a). In the analyses (c) and (e), note that the category for the transitive verb *eats* is set to (S\NP)\((S\NP)\NP), not (S\NP)\NP. The difference is that the subject NP *Kim* is also type-raised in (e). Therefore, in the analysis (c), two times of *backward functional application*s are applied. After the *backward functional application* in (2b) is applied to combine a transitive *eats* and the object NP *pizza*, the *backward functional application* in (2b) is applied one more time to combine the subject NP *Kim* and the VP *eats pizza*. In the analysis (e), after the *backward functional application* in (2b) is applied to combine a transitive *eats* and the object NP *pizza*,

the subject NP *Kim* is *forward type raised* by (3a). Then, the *forward functional application* in (2a) is applied to combine the subject NP *Kim* and the VP *eats pizza*.

Though these five CCG analyses may be possible for even the simple sentence in (1), to produce all of the possible analyses during the parsing processes is unreasonable, since it increases the time and space complexity in the parsing processes. The most reasonable solution to this problem is to produce just one and simplest parse tree for each sentence unless the sentence itself does not contain structural ambiguity.[2)]

In order to solve this problem, Beavers (2002, 2004) introduced the type Normal Form *nf* and set the type hierarchy of this type as in Figure 2 (2002, p. 101). The explanations of each type in the type hierarchy are provided in Table 1 (2002, p. 101-102).

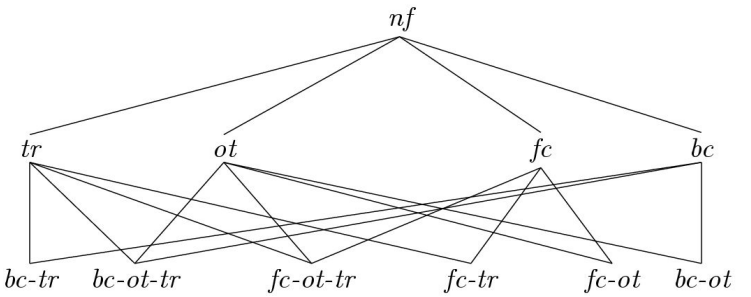


Figure 2. Type Hierarchy for the type *nf*

2) Note that the LKB system takes a chart parsing algorithm. Accordingly, all the possible structures are generated during the parsing processes. The most reasonable and desirable way to reduce the unnecessary syntactic ambiguity during the parsing processes is to impose some constraints on type hierarchies or (grammatical) rule system, since the LKB system is based on typed feature structure formalism. Beavers (2002, 2004) took this approach, and this paper also takes the same approach. However, the constraints themselves have to be different because Korean is different from English.

Table 1. Explanations of Each Type in *nf*

feature	meaning
bc	output of <B
fc	output of >B
ot	output of > or < or a lexical item
tr	output of >T or <T
bc-tr	output of <B or >T or <T
fc-tr	output of >B or >T or <T
bc-ot	output of <B or > or < or a lexical item
fc-ot	output of >B or > or < or a lexical item
bc-ot-tr	output of <B, > or <, >T or <T, or a lexical item
fc-ot-tr	output of >B, > or <, >T or <T, or a lexical item

The type *nf* encodes the information on what kind of category combinatorics is applied when one constituent combines another constituent(s), and it imposes some constraints on the possible types of category combinatorics in the next step. The Attribute-Value Matrix (AVM) for the type *feature-struct* contains this information as in Figure 3 (2002, p. 101).

$$\begin{bmatrix} \textit{feature-struct} \\ \text{NF } \textit{nf} \\ \text{ORTH } \textit{*diff-list-of-strings*} \\ \text{SS } \textit{synsem} \\ \text{DTRS } \textit{list-of-signs} \end{bmatrix}$$

Figure 3. AVM for the Type *feature-struct*

Note that the type *feature-struct* contains the attribute NF, and its value has to be *nf* which has to be one of the types in Figure 2.

Based on the type hierarchy in Figure 1 and the AVM in Feature 2, Beavers (2002, p. 102) imposed the constraints on each category combinatorics as in Figure 4.[3)]

3) The category combinatorics *functional substitution* is not included here for convenience. Here, P-DTR is the attribute for the primary daughter and S-DTR is for the subordinate

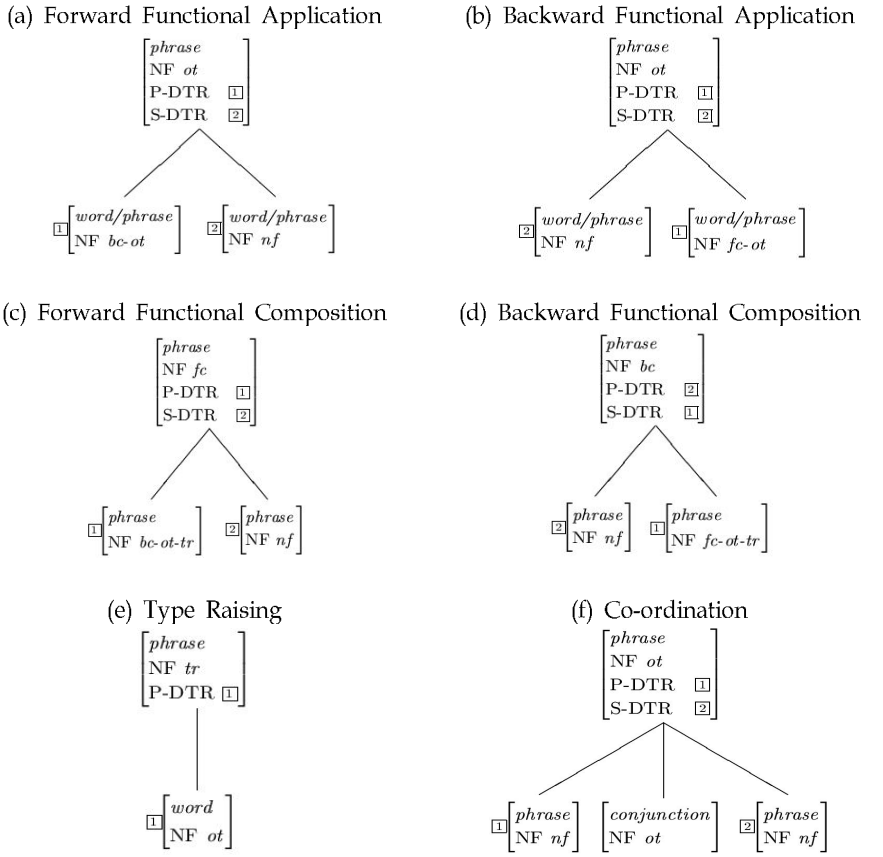


Figure 4. Category Combinatorics and the Type *nf*

By adopting the type *nf* and its type hierarchy, he tried to reduce the syntactic ambiguity of English parse trees (in his English TCCG system) within the reasonable time and space.

Now, let's see how the constraints in Figure 4 correctly select the simplest analysis in (a) of Figure 1. First of all, Beavers (2002, 2004) sets the category of a transitive (S\NP)\NP. A transitive verb cannot be type raised, and its category cannot be changed. Therefore, the analyses (c) and (e) cannot be generated during the parsing processes. In the analysis (d), there is no problem when a

(secondary) daughter.

transitive *eats* and the object NP *pizza* are combined by the *forward functional application* in (2a). A problem occurs when the type-raised subject NP *Kim* combines with a VP *eats pizza* once again by a *forward functional application* in (3a). According to the constraints in (a) of Figure 4, the NF value of the P-DTR for the *forward functional application* is *bc-ot*. The NF value of the type-raised subject NP *Kim* is *tr* as you can observe (e) of Figure 4. The NF value *tr* does not fit into the constraints of P-DTR of *forward functional application*, and this operation does not occur during the parsing processes. Now, let's see the analysis in (b). There is no problem when the type-raised subject NP *Kim* combines with a transitive *eats* by a *forward functional composition* in (3a). The problem occurs when this constituent combine with the object NP *pizza* by the *forward functional application* in (2a). According to the constraints in (a) of Figure 4, the NF value of the P-DTR for the *forward functional application* is *bc-ot*. The NF value of the constituent *Kim eats* is *fc* as you can observe (c) of Figure 4. The NF value *fc* does not fit into the constraints of P-DTR of *forward functional application*, and this operation does not occur during the parsing processes.

By adopting the type *nf*, its type hierarchy, and the constraints on the category combinatorics, Beavers succeeded to reduce the syntactic ambiguity in his English TCCG system.

3. Syntactic Ambiguity in the Korean TCCG System

3.1. Problems

Though Beavers' approach is successful to reduce syntactic ambiguity in the English TCCG system, his type hierarchy of *nf* doesn't fit into the K-TCCG system (Lee, 2010a) because of head parameters. Note that English is a head initial language, whereas Korean is a head-final language. Since Korean is a head-final language, an NP may combine with another NP with *type raising* before it combines with a predicate, and it makes the syntactic ambiguity problem even in simple sentences.

Let's see a simple example sentence in (5).

(5) *Chelsoo-ka* *Younghee-lul* *po-ass-ta.*
Cheloo.Nom Younghee.ACC see.PAST.DECL
'Chelsoo saw Younghee.'

For this simple sentence (5), the following four different CCG analyses are possible, even though we exclude the cases where the category of a transitive is changed.

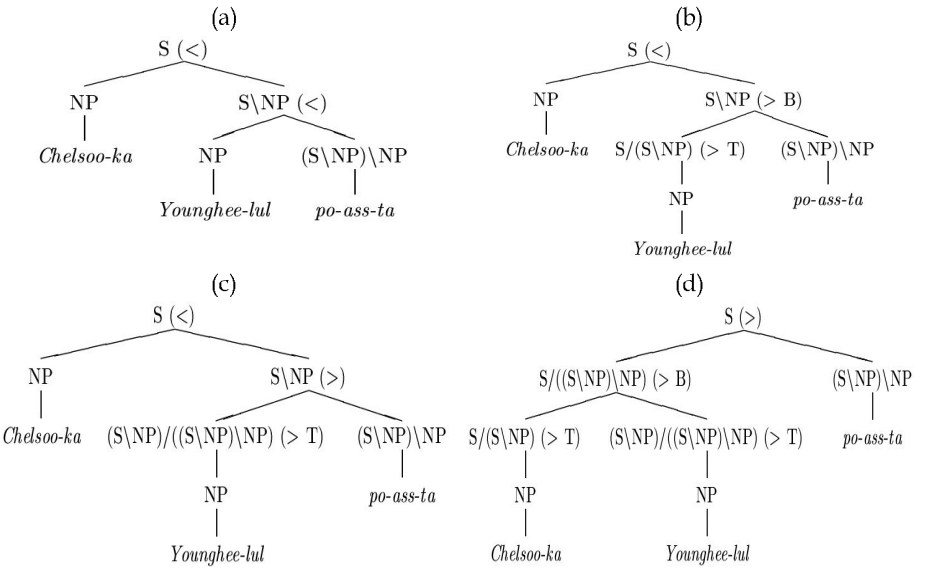


Figure 5. Four Possible CCG Analyses for the English Sentence in (5)

These four analyses are constructed as follows. in the analysis (s), two times of *backward functional applications* are applied. After the *backward functional application* in (2b) is applied to combine a transitive *po-ass-ta* and the object NP *Younghee-lul*, the *backward functional application* in (2b) is applied one more time to combine the subject NP *Chelsoo-ka* and the VP *Younghee-lul po-ass-ta*. The operation *type raising* is involved in other analyses. In the analyses (b) and (c), a *forward functional composition* in (3a) is involved in the object NP *Younghee-lul*. In the analysis (b), the type-raised object NP *Younghee-lul* and the transitive verb *po-ass-ta* are combined by the *forward functional composition* in (3a). Then, the

subject NP *Chelsoo-ka* combines with a VP *Younghee-lul po-ass-ta* by a *backward functional application* in (2a). In the analysis (c), the type-raised object NP *Younghee-lul* and the transitive verb *po-ass-ta* are combined by the *forward functional application* in (2a). Then, the subject NP *Chelsoo-ka* combines with a VP *Younghee-lul po-ass-ta* by a *backward functional application* in (2b). In the analysis (d), both the subject NP *Chelsoo-ka* and the object NP *Younghee-lul* are type-raised by the *forward type raising* in (4a). Then, two NPs are combined by the *forward functional composition* in (3a). Then, this constituent combines with the transitive verb *po-ass-ta* by a *forward functional application* in (2a).

Even though the above four CCG analyses are possible for the simple sentence in (5), the semantic interpretation of this sentence is identical, which can computationally be implemented as in Figure 6.[4)]

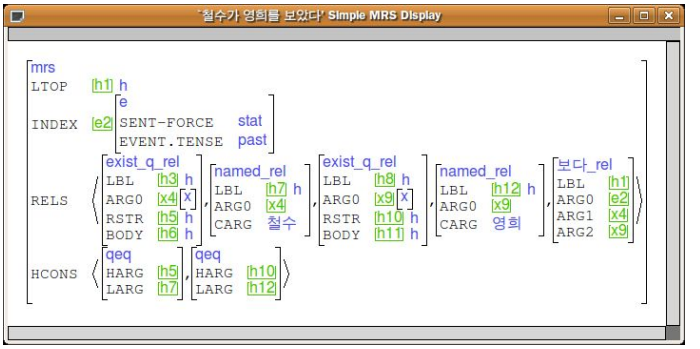


Figure 6. Semantic Interpretation of the Sentence (5)

Therefore, it is desirable to produce one and simplest parse tree in (a) for this

4) As you may find in Figure 6, the K-TCCG system is similar to the second version of the Korean Resource Grammar (KRG2; Song et al., 2010), rather than the first version of the Korean Resource Grammar (KRG1; Kim and Yang, 2003). However, the K-TCCG system is different from the KRG2 in the following two ways. First, two systems use different kind of syntactic rules. The KRG systems make use of the syntactic rules in Head-driven Phrase Structure Grammar (HPSG; Pollard & Sag (1994), Sag and Wasow (1999), and Sag et al. (2003)), whereas the K-TCCG system makes use of the category combinatorics in CCG. Second, the grammar matrix (Bender et al., 2002) was incorporated in the KRG2 system, while the K-CCG system contains no element in the grammar matrix. However, the type hierarchies of the Lexicon and lexical rules are similar in these two systems, though they are not exactly identical.

sentence unless the sentence itself contains structural ambiguity.

Note that the constraints in Figure 4 exclude two possible analyses, the analysis (c) and (d), among these four possible analyses. In the analysis (c), the problem occurs when the type-raised object NP *Younghee-lul* and the transitive verb *po-ass-ta* are combined by the *forward functional application* in (2a). According to the constraints in (a) of Figure 4, the NF value of the P-DTR for the *forward functional application* is *bc-ot*. The NF value of the type-raised object NP *Younghee-lul* is *tr* as you can observe (e) of Figure 4. The NF value *tr* does not fit into the constraints of P-DTR of *forward functional application*, and this operation does not occur during the parsing processes. In the analysis (d), there is no problem when the subject NP *Chelsoo-ka* combines with the object NP *Younghee-lul* by the *forward functional composition* in (3a). The problem occurs when this constituents combines with the transitive verb *po-ass-ta* by a *forward functional application* in (2a). According to the constraints in (a) of Figure 4, the NF value of the P-DTR for the *forward functional application* is *bc-ot*. The NF value of the constituent *Kim eats* is *fc* as you can observe (c) of Figure 4. The NF value *fc* does not fit into the constraints of P-DTR of *forward functional application*, and this operation does not occur during the parsing processes. The other two analyses are possible for the Korean sentence in (5).

However, just excluding the analysis (d) of Figure 5 from the analyses is not a welcome result since this strategy may raise another problem, which we may encounter in the computational implementation of the RNR constructions. (6) illustrates an example of the Korean RNR constructions, and its CCG analysis is shown in Figure 7. (Lee, 2010b, p. 28-29)

(6) *Chelsoo-ka* *Younghee-lul* *kuliko* *Minsoo-ka*
Chelsoo.NOM Younghee.ACC and Minsoo.NOM
Sunhee-lul *po-ass-ta.*
Sunhee.ACC see.PAST.DECL
'Chelsoo saw Younghee, and Minsoo saw Sunhee.'

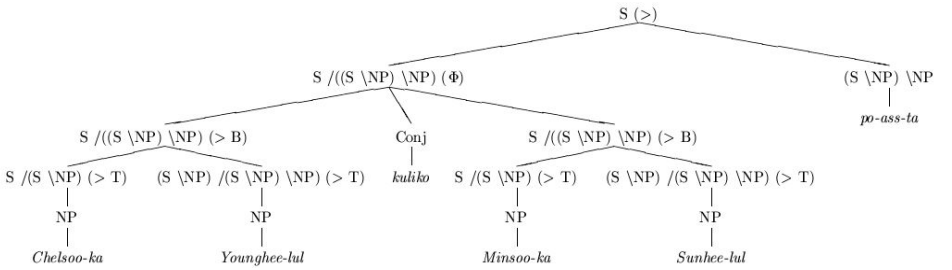


Figure 7. A CCG Analyses for the English Sentence in (6)

The analysis in Figure 7 contains the category combinatorics which is similar to those in the analysis (d) of Figure 5. However, this analysis is well-formed in Korean. Accordingly, in the K-TCCG system, we have to make the analysis (d) of Figure 5 impossible while that of the Figure 7 possible. That is what we have to do in this paper.

3.2. Modification of the Type Hierarchy

In order to solve this problem, the K-TCCG system takes the type hierarchy of the type *nf* as in Figure 8. The explanations of each type in the type hierarchy are provided in Table 2. As you can observe, the type *co* and *bc-ot-co* are added to the hierarchy.

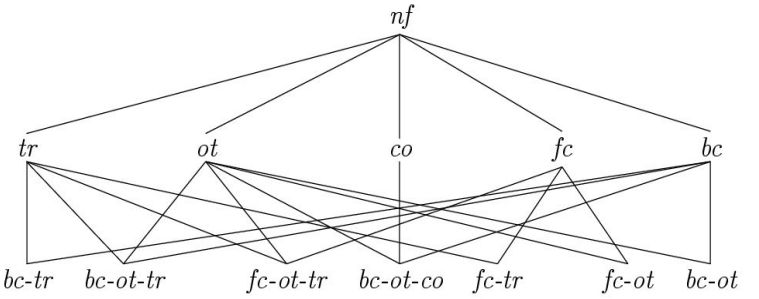


Figure 8. Type Hierarchy for the type *nf* in the K-TCCG System

Table 2. Explanations of Each Type in *nf*

feature	meaning
bc	output of <B
fc	output of >B
ot	output of > or < or a lexical item
tr	output of >T or <T
co	output of Φ
bc-tr	output of <B or >T or <T
fc-tr	output of >B or >T or <T
bc-ot	output of <B or > or < or a lexical item
fc-ot	output of >B or > or < or a lexical item
bc-ot-tr	output of <B, > or <, >T or <T, or a lexical item
fc-ot-tr	output of >B, > or <, >T or <T, or a lexical item
bc-ot-co	output of <B, or Φ, or > or < or a lexical item

As in Beavers' TCCG system, the type *nf* encodes the information on what kind of category combinatorics is applied when one constituent combines another constituent(s), and it imposes some constraints on the possible types of category combinatorics in the next step. The AVM for the type *feature-struct* is modified as in Figure 9.

$$\begin{bmatrix} \textit{feature-struct} \\ \text{NF } \textit{nf} \\ \text{PHON } \textit{*dlist*} \\ \text{SYN } \textit{syn} \\ \text{SEM } \textit{sem} \\ \text{DTRS } \textit{*list*} \end{bmatrix}$$

Figure 9. AVM for the Type *feature-struct*

Note that the type *feature-struct* contains the attribute NF, and its value has to be *nf*, one of the types in Figure 8.

Based on the type hierarchy in Figure 8 and the AVM in Feature 9, the K-TCCG system imposes the constraints on each category combinatorics as in Figure 10.

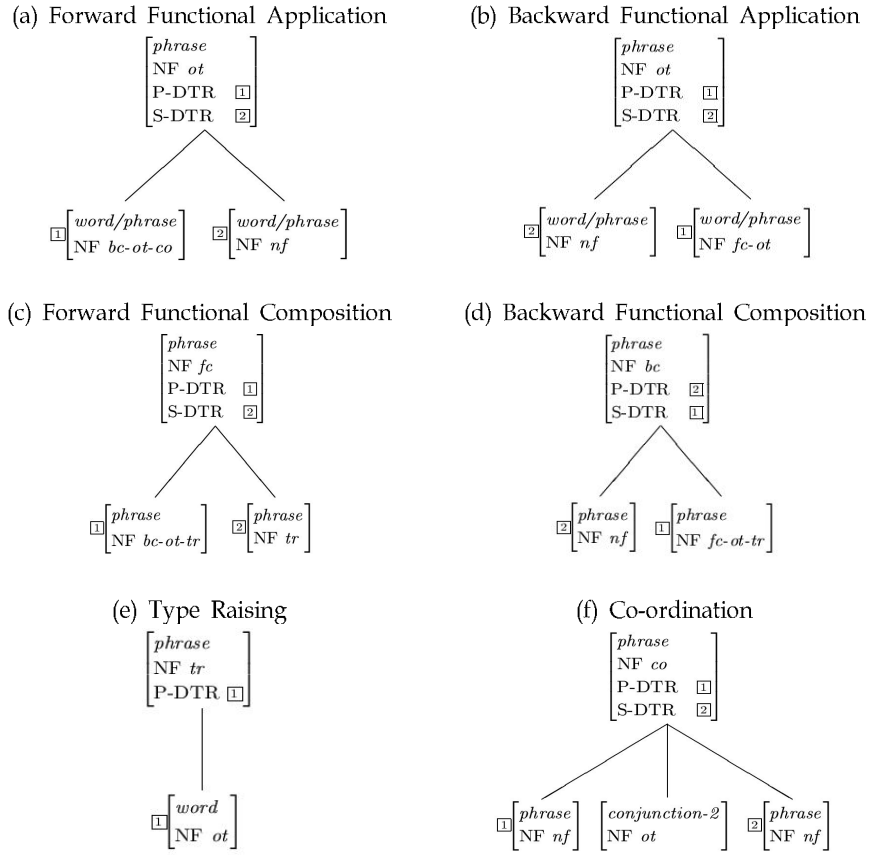


Figure 10. Category Combinatorics and the Type *nf* in the K-TCCG System

Here, the constraints on (a) *forward functional application*, (c) *forward functional composition*, and (f) *co-ordination* are changed.

3.3. Normal Form and Syntactic Ambiguity in the Korean TCCG System

Now, let's see how the modified constraints in Figure 10 can correctly select the simplest and the most desirable parse tree among the four possible CCG analyses in Figure 5. Figure 11-Figure 14 demonstrate how these four analyses can be handled with the modified type hierarchy and the constraints on the category combinatorics. In each analysis, the attribute NF is added in addition to

SYN|CAT which encoded the category information of each node.

The analysis in Figure 11 corresponds to the analysis (a) of Figure 5.

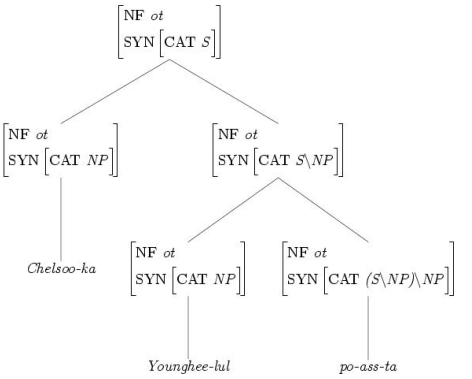


Figure 11. Analysis (a) and the Type *nf*

In this analysis, two times of *backward functional application* are applied. As you may observe in Figure 11, the NF value *ot* of *po-ass-ta* satisfies the constraints *fc-ot* in (b) of Figure 10. Once again, the NF value *ot* of *Younghee-lul po-ass-ta* satisfies the constraints in (b) of Figure 10. Accordingly, this analysis tree is available during the parsing processes.

The analysis in Figure 12 corresponds to the analysis (b) of Figure 5.

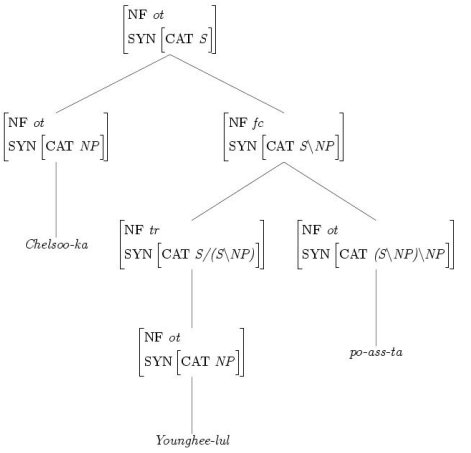


Figure 12. Analysis (b) and the Type *nf*

In this analysis, *forward type raising*, *forward functional composition*, and *backward functional application* are applied. The problem occurs when type-raised object NP *Younghee-lul* combines with the transitive verb *po-ass-ta* by the *forward functional composition*. The NF value *ot* of *po-ass-ta* does not satisfy the constraints in (c) of Figure 10, since the constraint of the *forward functional composition* says that the NF value of the S-DTR (subordinate (secondary) daughter) has to be *tr*. Accordingly, this analysis tree is not available during the parsing processes.

The analysis in Figure 13 corresponds to the analysis (c) of Figure 5.

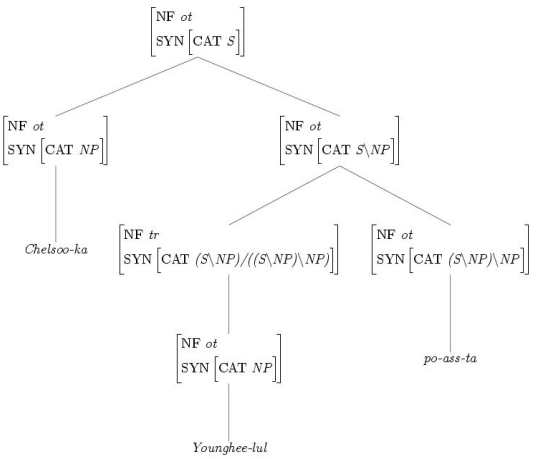


Figure 13. Analysis (c) and the Type *nf*

In this analysis, *forward type raising*, *forward functional application*, and *backward functional application* are applied. The problem occurs when type-raised object NP *Younghee-lul* combines with the transitive verb *po-ass-ta* by the *forward functional application*. The NF value *tr* of *Younghee-lul* does not satisfy the constraints in (c) of Figure 10, since the constraint of the *forward functional composition* says that the NF value of the primary daughter (P-DTR) has to be *bc-ot-co*. Accordingly, this analysis tree is not available during the parsing processes.

Now, let's see the final analysis. The analysis in Figure 14 corresponds to the analysis (d) of Figure 5.

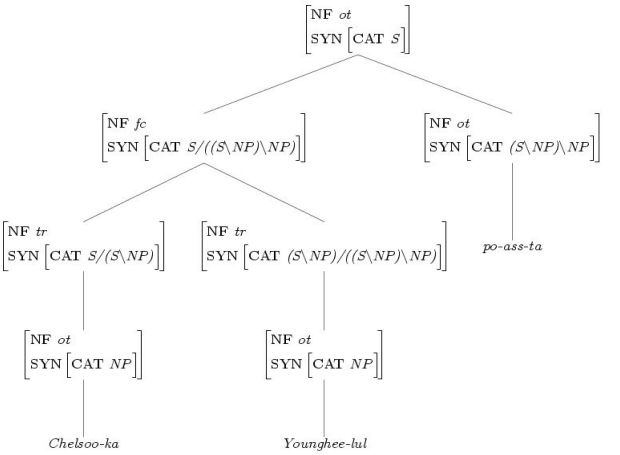


Figure 14. Analysis (d) and the Type *nf*

In this analysis, two times of *forward type raising*, *forward functional application*, and *backward functional application* are applied. The problem occurs when the composed NP *Chelsoo-ka Younghee-lul* combines with the transitive verb *po-ass-ta* by the *forward functional application*. The NF value *fc* of *Chelsoo-ka Younghee-lul* does not satisfy the constraints in (a) of Figure 10, since the constraint of the *forward functional application* says that the NF value of the primary daughter (P-DTR) has to be *bc-ot-co*. Accordingly, this analysis tree is not available during the parsing processes.

3.4. RNR Constructions and Normal Form

Now, let's see how the modified constraints in Figure 10 can correctly analyze the RNR construction in Figure 7. Figure 15 demonstrate how the RNR construction can be handled with the modified constraints.

The analysis proceeds as follows. First, two NPs *Chelsoo-ka* and *Younghee-lul* are type raised by the *forward type raising* in (4a). Then, two constituents are combined by the *forward functional composition* in (3a). The NF value *tr* in both constituents satisfies the constraints in (c) of Figure 10. Likewise, two NPs *Minsoo-ka* and *Sunhee-lul* are type raised by the *forward type raising* in (4a), and two constituents are combined by the *forward functional composition* in (3a). Then, two conjuncts are co-ordinated. Here, the constraints in (f) of Figure 10 are

applied. Note that the NF value of the co-ordinated constituents is *co*. Finally, the co-ordinated constituents are combined with the transitive verb *po-ass-ta* by the *forward functional application*. The constraint of the *forward functional application* says that the NF value of P-DTR (the primary daughter) has to be *bc-ot-co*, and the NF value of the co-ordinated constituents is *co*. Accordingly, this combination satisfies the constraints in (a) of Figure 10, and the this analysis tree is available during the parsing processes.

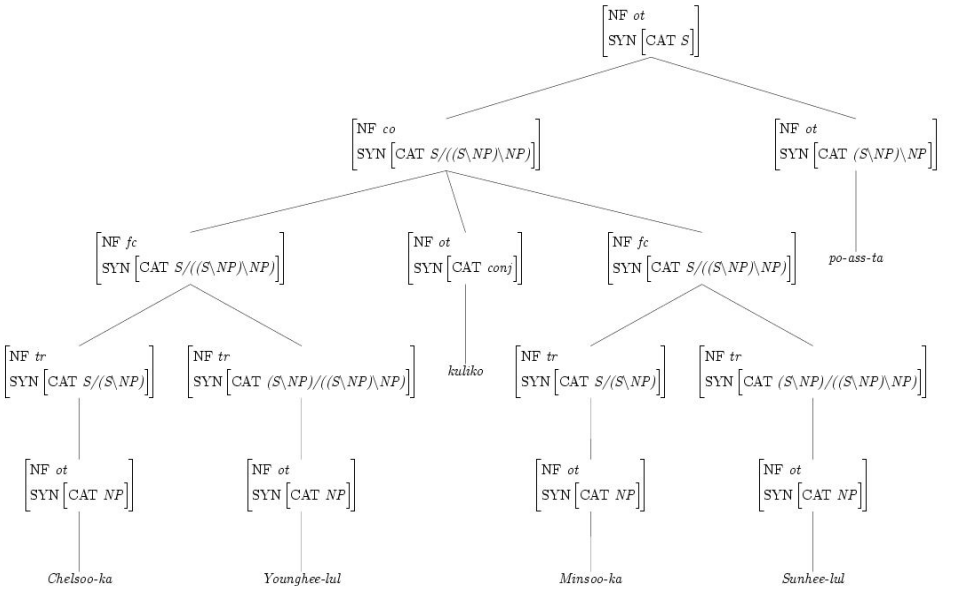


Figure 15. An RNR Construction and the Type *nf*

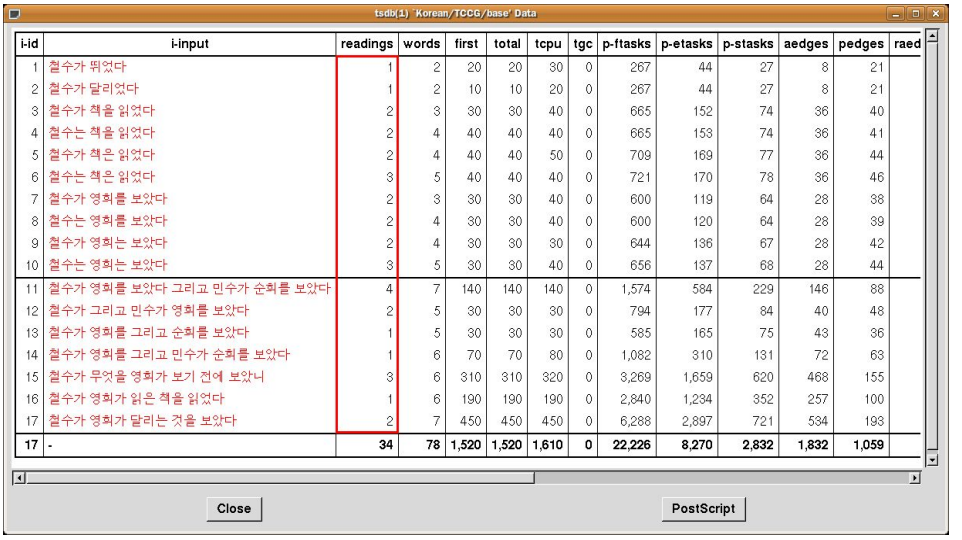
4. Results of Implementation

In order to check if the modified type hierarchy of *nf* and the constraints on category combinatorics properly work, it is necessary to examine the implementation results. Figure 16 demonstrates a [incr tsdb()] screenshot which was taken before the modification. Note that the number of readings for each sentence is more than one in most of the sentences. For even simple sentences, note that more than one reading is available. For example, for the sentence (5)

(i-id 7 in Figure 16) has 2 readings, which correspond to (a) and (b) in Figure 5.

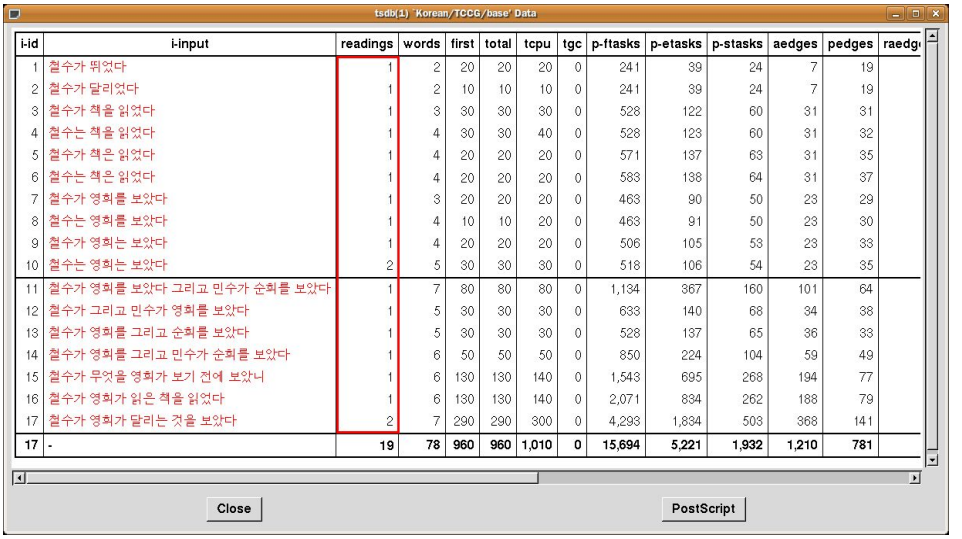
On the other hand, Figure 17 demonstrates the [incr tsdb()] screenshot after the modification. Note the number of readings for each sentence. For most simple sentences, note that only one reading is available. For example, for the sentence (5) (i-id 7 in Figure 16) has only one reading.

If you compare Figure 16 and Figure 17, you may find that the number of readings is significantly reduced in Figure 17, compared with the number of readings in Figure 16. From these results, we may find that the modified type hierarchy for *nf* correctly select the simplest parse tree for each sentence without affecting other parts of the grammar.



i-id	i-input	readings	words	first	total	tcpu	tgc	p-ftasks	p-etasks	p-stasks	aedges	pedges	raed
1	철수가 뛰었다	1	2	20	20	30	0	267	44	27	8	21	
2	철수가 달리었다	1	2	10	10	20	0	267	44	27	8	21	
3	철수가 책을 읽었다	2	3	30	30	40	0	665	152	74	36	40	
4	철수는 책을 읽었다	2	4	40	40	40	0	665	153	74	36	41	
5	철수가 책은 읽었다	2	4	40	40	50	0	709	169	77	36	44	
6	철수는 책은 읽었다	3	5	40	40	40	0	721	170	78	36	46	
7	철수가 영희를 보았다	2	3	30	30	40	0	600	119	64	28	38	
8	철수는 영희를 보았다	2	4	30	30	40	0	600	120	64	28	39	
9	철수가 영희는 보았다	2	4	30	30	30	0	644	136	67	28	42	
10	철수는 영희는 보았다	3	5	30	30	40	0	656	137	68	28	44	
11	철수가 영희를 보았다 그리고 민수가 순희를 보았다	4	7	140	140	140	0	1,574	584	229	146	88	
12	철수가 그리고 민수가 영희를 보았다	2	5	30	30	30	0	794	177	84	40	48	
13	철수가 영희를 그리고 순희를 보았다	1	5	30	30	30	0	585	165	75	43	36	
14	철수가 영희를 그리고 민수가 순희를 보았다	1	6	70	70	80	0	1,082	310	131	72	63	
15	철수가 무엇을 영희가 보기 전에 보았니	3	6	310	310	320	0	3,269	1,659	620	468	155	
16	철수가 영희가 읽은 책을 읽었다	1	6	190	190	190	0	2,840	1,234	352	257	100	
17	철수가 영희가 달리는 것을 보았다	2	7	450	450	450	0	6,288	2,897	721	534	193	
17	-	**34**	**78**	**1,520**	**1,520**	**1,610**	**0**	**22,226**	**8,270**	**2,832**	**1,832**	**1,059**	

Figure 16. [incr tsdb()] Screenshot (Before the Modification)



i-id	i-input	readings	words	first	total	tcpu	tgc	p-ftasks	p-etasks	p-stasks	aedges	pedges	raedg
1	철수가 뛰었다	1	2	20	20	20	0	241	39	24	7	19	
2	철수가 달리었다	1	2	10	10	10	0	241	39	24	7	19	
3	철수가 책을 읽었다	1	3	30	30	30	0	528	122	60	31	31	
4	철수는 책을 읽었다	1	4	30	30	40	0	528	123	60	31	32	
5	철수가 책은 읽었다	1	4	20	20	20	0	571	137	63	31	35	
6	철수는 책은 읽었다	1	4	20	20	20	0	583	138	64	31	37	
7	철수가 영희를 보았다	1	3	20	20	20	0	463	90	50	23	29	
8	철수는 영희를 보았다	1	4	10	10	20	0	463	91	50	23	30	
9	철수가 영희는 보았다	1	4	20	20	20	0	506	105	53	23	33	
10	철수는 영희는 보았다	2	5	30	30	30	0	518	106	54	23	35	
11	철수가 영희를 보았다 그리고 민수가 순희를 보았다	1	7	80	80	80	0	1,134	367	160	101	64	
12	철수가 그리고 민수가 영희를 보았다	1	5	30	30	30	0	633	140	68	34	38	
13	철수가 영희를 그리고 순희를 보았다	1	5	30	30	30	0	528	137	65	36	33	
14	철수가 영희를 그리고 민수가 순희를 보았다	1	6	50	50	50	0	850	224	104	59	49	
15	철수가 무엇을 영희가 보기 전에 보았니	1	6	130	130	140	0	1,543	695	268	194	77	
16	철수가 영희가 읽은 책을 읽었다	1	6	130	130	140	0	2,071	834	262	188	79	
17	철수가 영희가 달리는 것을 보았다	2	7	290	290	300	0	4,293	1,834	503	368	141	
17	-	**19**	**78**	**960**	**960**	**1,010**	**0**	**15,694**	**5,221**	**1,932**	**1,210**	**781**	

Figure 17. [incr tsdb()] Screenshot (After the Modification)

5. Conclusion

This paper tried to solve the efficiency problem which was raised by syntactic ambiguity in parse trees. In order to handle this problem, Beavers (2002, 2004) introduced the type Normal Form *nf* and its type hierarchy. The type *nf* encodes the information on what kind of category combinatorics is applied when one constituent combines another constituent, and it also imposes some constraints on the possible types of category combinatorics in the next step. By adopting the type *nf* and its type hierarchy, he succeeded to reduce the syntactic ambiguity of English parse trees within the reasonable time and space.

However, Beavers' type hierarchy of *nf* doesn't fit into the K-TCCG (Korean TCCG) system because the head parameters work between two languages. Since Korean is a head-final language, an NP may combine with another NPs with *type raising* before it combines with a predicate, and it causes the syntactic ambiguity problem even in the simple sentences.

In order to solve this syntactic ambiguity problem in the K-TCCG system, this paper modifies the type hierarchy of *nf* so that the K-TCCG system can properly implement various syntactic phenomena in the Korean language. One

of the problematic constructions in Korean was the RNR constructions, where the operations *type raising* and *forward functional application* are used. However, in the modified type hierarchy of *nf*, we found that the RNR constructions as well as simple sentences can be effectively implemented without any syntactic ambiguity in parse trees. We also found that the modified type hierarchy of *nf* and the constraints on the category combinatorics properly work through the examination of the computational implementation.

References

Baldridge, J. (2002). *Resource-sensitive combinatory categorial grammar.* Unpublished doctoral dissertation. University of Edinburg. Edinburg, United Kingdom.

Beavers, J. (2002). A CCG implementation for the LKB. LinGO Working Paper #2002-8. Stanford, CA: CSLI.

Beavers, J. (2004). Type-inherited combinatory categorial grammar. In *Proceedings of the 20th International Conference on Computational Linguistics.* Article No. 57.

Bender, E. (2008). Grammar engineering for linguistic hypothesis testing. In S. Gaylord, S. Hilderbrand, H. Lyu, A. Palmer & E. Ponvert (Eds.), *Texas linguistics society 10: computational linguistics for less-studied languages* (pp. 16-36). Stanford, CA: CSLI.

Bender, E. & Flickinger, D. (2005). Rapid prototyping of scalable grammars: towards modularity in extensions to a language-independent core. In *Proceedings of the Second International Joint Conference on Natural Language Processing* (IJCNLP-05), Poster/Demos, Jeju Island, Korea.

Bender, E., Flickinger, D. & Oepen, S. (2002). The Grammar matrix: an open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In Carroll, J., Oostdijk, N. & Sutcliffe, R. (Eds.), *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics,* 8-14. Teipei, Taiwan.

Bond, F., Oepen, S., Siegel, M., Copestake, A. & Flickinger, D. (2005). Open source machine translation with DELPH-IN. *Open Source Machine Translation: Workshop at MT Summit*, X, 15-22.

Callmeier, U. (2001). Efficient parsing with large-scale unification grammars. Unpublished MA thesis. Universitat Saarlandes.

Carpenter, B. (1992). Lexical and unary rules in categorial grammar. In B. Levin, (Ed.), *Formal Grammar: Theory and Implementation, Vol. 2 of Vancouver studies in cognitive science* (pp. 168-242). Oxford: Oxford University Press.

Copestake, A. (2002). *Implementing Typed Feature Structure Grammar*. Stanford, CA: CSLI.

Copestake, A., Flickinger, D., Malouf, B., Riehemann, S. & Sag, I. (1995). Translation using minimal recursion semantics. In *Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation* (TMI95), 15-32.

Hockenmaier, J. Bierner, G. & Baldridge, J. (2001). Extending the coverage of a CCG system. *Research on Language and Computation*, 2(2), 165-208.

Kim, Jong-bok & Yang, Jaehyung. 2003. Korean Phrase Structure Grammar and Its Implementations into the LKB System. In *Proceedings of the 17th Pacific Asia Conference on Language, Information and Computation* (PACLIC-17), 88-97.

Lee, Yong-hun. (2010a). Type-inherited combinatory categorial grammar for a Korean fragment: with reference to English. *Studies in English Language & Literature*, 36(3), 275-292.

Lee, Yong-hun. (2010b). Implementation of RNR constructions in the Korean TCCG system, *The Linguistic Association of Korea Journal*, 18(2), 27-46.

Oepen, S. (2002). Competence and Performance Profiling for Constraint-based Grammars: A New Methodology, Toolkit, and Applications. Doctoral dissertation. Universitat Saarlandes.

Pollard, C. & Sag, I. (1994). *Head-driven Phrase Structure Grammar*. Chicago: University of Chicago Press.

Sag, I. & Wasow, T. (1999). *Syntactic Theory: A Formal Introduction*. First Edition. Stanford, CA: CSLI.

Sag, I., Wasow, T. & Bender, E. (2003). *Syntactic Theory: A Formal Introduction*. Second Edition. Stanford, CA: CSLI.

Song, Sanghoun, Kim, Jong-Bok, Bond, Francis & Yang, Jaehyung. (2010). Development of the Korean Resource Grammar: towards grammar customization. In *Proceedings of the Eighth Workshop on Asian Language Resources*, 144-152.

Steedman, M. (1996). *Surface structure and interpretation.* Cambridge, MA: MIT Press.

Steedman, M. (2000). *The syntactic process.* Cambridge, MA: MIT Press.

Vallavicencio, A. (2001). The acquisition of a unification-based generalized categorial grammar. Technical report. Cambridge University.

Yong-hun Lee
Department of English Language & Literature
Chungnam National University
220 Gung-dong, Yuseng-gu,
Daejeon 305-764, Korea
Phone: 82-42-821-5331

Department of English Language & Literature
Hannam University
133 Ojeong-dong, Daedeok-gu,
Daejeon 306-791, Korea
Phone: 82-42-629-7320
Email: ylee@cnu.ac.kr

Received on 30 March, 2011
Revised on 30 April, 2011
Accepted on 30 April, 2011