Recovering Empty Arguments in Korean

H.S. Park, D. Egedi, M. Palmer

Institute for Research in Cognitive Science
University of Pennsylvania
Philadelphia, PA 19104-6228, USA

{hspark, egedi, mpalmer}@linc.cis.upenn.edu

Abstract

This paper looks at empty arguments in Korean, and the constraints that can be used in recovering their meaning, or referent. We look at scrambling to provide insights into the process by which empty arguments can be recovered, and provide a computational algorithm based on local and global information that takes advantage of the semantic restrictions placed by the verb. We look at recovering empty arguments at both the sentential and dialog levels.

1 Introduction

One of the most problematic elements in parsing Korean text is the widespread use of empty arguments. 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. There are linguistic theories [1] that account for this phenomena, but they lack a computational component that would be useful when parsing Korean text. We begin the paper with a discussion of scrambling, and argue that the problem of resolving empty arguments in Korean is closely related to the problem of matching arguments in scrambled Korean text. We then present a computational method for resolving the references of empty arguments based on a discourse model with a local stack and a global ordered list.

2 Characteristics of Korean Arguments

2.1 Scrambling

A basic characteristic 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. As an example, consider the verb *sayngkakhanta* (*think*). It subcategorizes for a subject and a clausal object that ends with the complementizer *-ko*. A TAG[1] representation of the tree is given in Figure 1.

We would predict that scrambling could occur between the clausal argument (C) and the subject (SP_0), as well as locally within the embedded clause (S_f). The canonical form of a sentence *Tom thinks that Jerry ate an apple* is given in sentence (1)[2]. The embedded clause is shown in brackets. Figure 2 shows the TAG representation of sentence (1)[3].

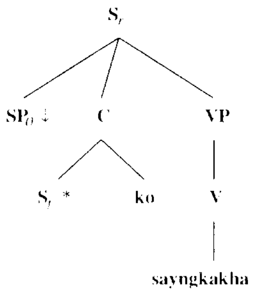

Figure 1: *Sayngkakha-(nta)* (*think*) tree

(1)	*Tom-i*	*[Jerry-ka*	*sakwa-lul*	*mekess-ta-ko]*	*sayngkakha-nta.*
	Tom-NOM	[Jerry-NOM	apple-ACC	ate-COMP]	think-PRES-DEC

Tom thinks that Jerry ate an apple. (Canonical Ordering)

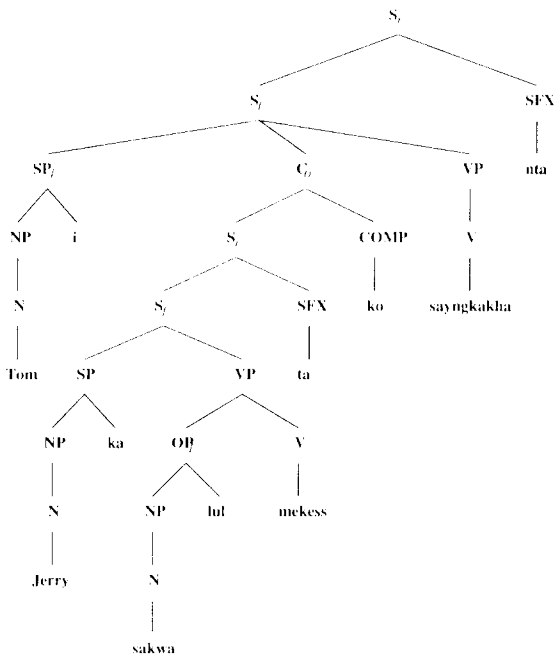

Figure 2: Canonical ordering of *Tom thinks that Jerry ate an apple*

Sentence (2) shows the same sentence with the subject NP and the embedded clause scrambled, while sentence (3) shows the elements within the embedded clause scrambled as well. Figure 3 gives the tree representation for sentence (3). Note that the embedded clause of *sayngkakha* (C_1) has been scrambled out of its original position. The coindexation of the **[trace]** feature between the C_0 and the C_1 indicates which site it corresponds to. Similarly, *sakwa-lul* has been scrambled out of its original position (OP_1) to the front of the sentence. Again, the coindexation of the **[trace]** feature indicates the correspondence between the two sites.

(2)	*[Jerry-ka*	*sakwa-lul*	*mekess-ta-ko]*	*Tom-i*	*sayngkakha-nta.*
	[Jerry-NOM	apple-ACC	ate-COMP]	Tom-NOM	think-PRES-DEC

Tom thinks that Jerry ate an apple. (Clause scrambled out of position)

(3)	*[sakwa-lul*	*Jerry-ka*	*mekess-ta-ko]*	*Tom-i*	*sayngkakha-nta.*
	[apple-ACC	Jerry-NOM	ate-COMP]	Tom-NOM	think-PRES-DEC

Tom thinks that Jerry ate an apple. (Scrambling within scrambled clause)

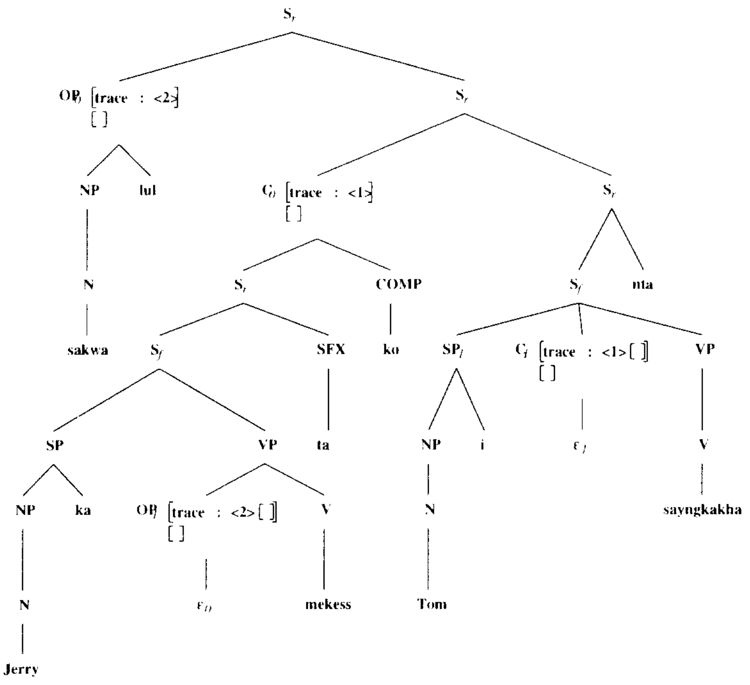

Figure 3: Scrambled order of *Tom thinks that Jerry ate an apple*

2.2 Long-Distance Scrambling

Korean also allows certain permutations of arguments which amount to **long-distance scrambling**, in which elements can be scrambled outside of their clausal boundaries [2][4]. Sentence (4) shows an example of this phenomena with *sakwa* (*apple*) scrambled outside of the embedded clause. A similar sentence is shown in sentence (5), with *Jerry* outside of the clausal boundaries. Although the sentence itself is well-formed, its meaning has changed; the subject arguments of the two verbs are reversed [3].

(4)	*sakwa-lul*	*Tom-i*	*[Jerry-ka*	*mekess-ta-ko]*	*sayngkakha-nta.*
	apple-ACC	Tom-NOM	[Jerry-NOM	ate-COMP]	think-PRES-DEC.

Tom thinks that Jerry ate an apple.

(5)	*sakwa-lul*	*Jerry-ka*	*Tom-i*	*[mekess-ta-ko]*	*sayngkakha-nta.*
	apple-ACC	Jerry-NOM	Tom-NOM	[ate-COMP]	think-PRES-DEC

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 (4) and (5) to the sentences in (6) and (7). The canonical form of the sentence is given in (6), while sentence (7) shows the long-distance scrambling version corresponding to sentence (5). The structure is the same in the two examples, with the embedded subject scrambled to the beginning of the sentence, but in sentence (7) the meaning does not change. The semantic restrictions on the subject of the verb *sayngkakhanta* (*think*) prohibit it from taking *swuhak* (*mathematics*) as its subject, while the semantics restrictions on *elyepta* (*be-difficult*) prohibit it from taking a human (*Tom*) as its subject. The NPs then, can be scrambled in any order.

(6)	*Tom-un*	*[swuhak-i*	*elyepta-ko]*	*sayngkakha-nta.*
	Tom-TOP	[mathematics-NOM	be-difficult-COMP]	think-PRES-DECL
	Tom thinks that mathematics is difficult.			

(7)	*swuhak-i*	*Tom-un*	*[elyepta-ko]*	*sayngkakha-nta.*
	mathematics-NOM	Tom-TOP	[be-difficult-COMP]	think-PRES-DECL
	Tom thinks that mathematics is difficult.			

2.3 Empty Arguments

The topic marker *-nun* is generally used in Korean to mark new information, and it precedes other information in the sentence. Once mentioned, lexical items that refer to that object may optionally drop 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. Consider the sentence in (8).

(8)	*Tom-un*	*i*	*mwuncey-nun*	*phwul swu-epta-ko*	*sayngkakha-nta.*
	Tom-TOP	this	problem-TOP	solve-NEG-COMP	think-PRES-DEC.
	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*[5]. There are several ways to interpret the use of the topic marker in this sentence. One way is to consider the topic marker *-nun* as ambiguously being a subject or object marker. The *nun* in *Tom-un* would be considered a subject marker, while the *nun* in *mwuncey-nun* would be considered an object marker. The language, however, provides unambiguous subject and object markers already, so it is unclear what the purpose of an ambiguous subject and object marker would be. It is also 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 swuepta* 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 4. 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 swuepta*[6] are all empty.

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 NPs optionally adjoin on. The first method (in which *-nun* is ambiguously a subject and

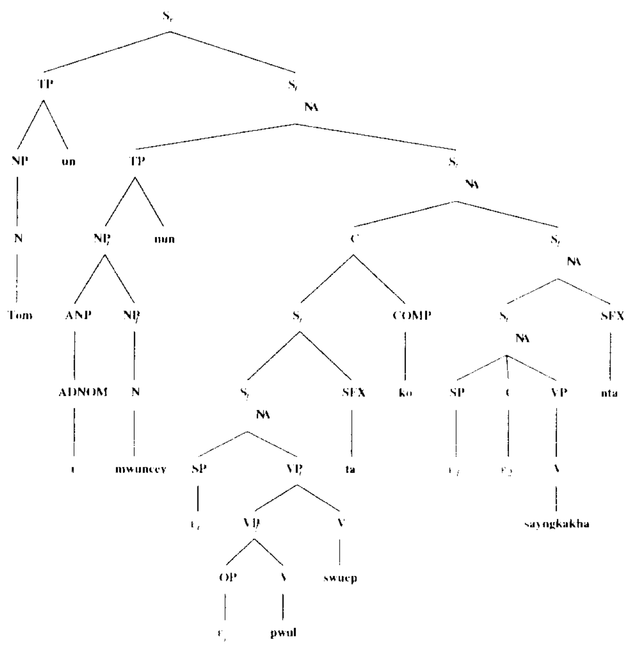

Figure 4: *Tom$_i$ thinks that he$_i$ can not solve this problem.*

object marker) 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 4), 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.

3 Recovering references

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.

The constraints on long-distance scrambling in Korean and the techniques used for recovering elided noun phrases in telegraphic English [4] and for resolving English anaphora [5, 6, 7] 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. We saw in sentences (4) and (5) that when *Jerry* and *Tom* are scrambled so that *Tom* is closer to the verb *mekessta* (*ate*), *Tom* becomes its subject, leaving *Jerry* as the subject of the verb *sayngkakhanta* (*think*).

We propose a general heuristic for recovering the referent of empty arguments as follows: Choose the closest topic NP that matches the semantic constraints on the elided arguments. We use a local stack (with look-ahead) as well as a global ordered list to select the appropriate referent for the empty argument. As each topic noun phrase is encountered, it is pushed onto the local stack, where is is available as a referent until it is popped off. After it has been popped off the stack, it is placed on a global ordered list and becomes available as a discourse referent. Clauses are processed from left to right across the sentence with the argument closest to the verb within the clause being processed first (that is, process the object before the subject).

3.1 An example

(9) *Tom-un i mwuncey-nun phwul swu-epta-ko sayngkakha-nta.*
Tom-TOP this problem-TOP solve-NEG-COMP think-PRES-DEC.
Tom_i thinks that he_i can not solve this problem.

Consider the sentence in (8) (reproduced as (9) above) again. Its graphical representation was given in Figure 4, 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. As each topic is encountered in the sentence, it is pushed onto the local stack, along with the semantic features associated with the lexical item. Figure 5a shows the state of the stack after both topic NPs have been pushed onto it.

The first clause encountered is the one headed by *phwul* (*solve*). Figure 5b shows the semantic feature constraints on its base tree. We process the argument closest to the verb (OP_0) 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[7]. The next empty argument is the subject of the main verb *sayngkakha* (*think*), whose tree is given in Figure 5c. 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*.

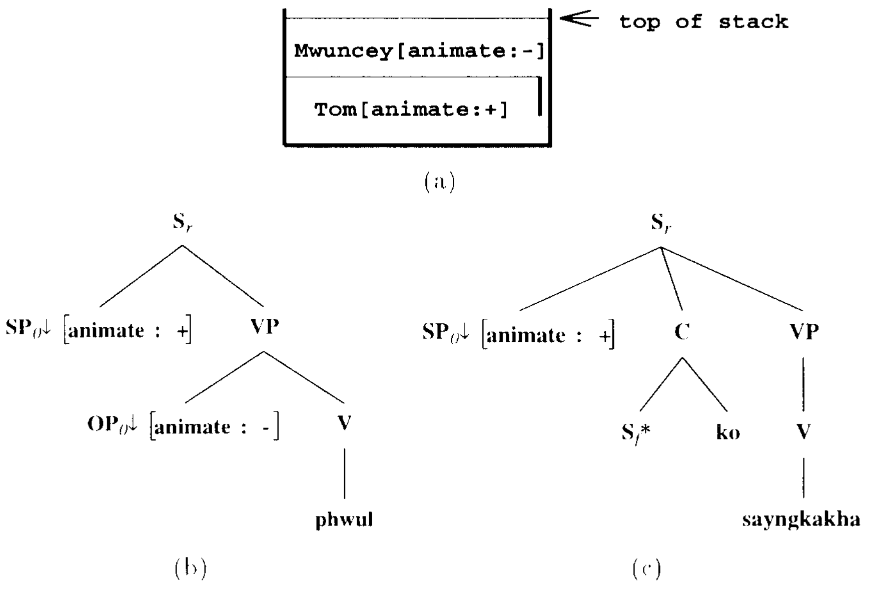

Figure 5: Recovering empty arguments using a stack

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 (10), then the semantic features, along with the look-ahead capabilities of the local stack, would be needed to correctly fill the empty arguments.

(10)	*i*	*mwunccy-nun*	*Tom-un*	*phwul swu-epta-ko*	*sayngkakha-nta.*
	this	problem-TOP	Tom-TOP	solve-NEG-COMP	think-PRES-DEC.

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, *mwunccy*. *Mwunccy* 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 (11).

(11)	**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.

3.2 Recovery at the Discourse Level

The example that was given in the previous section dealt with argument recovery at the sentential level. However, empty arguments in Korean can (and often do) get their referents from outside the sentence structure. Any sentence in which there are not enough NPs locally available must look to the discourse level for resolving the verb arguments.

In addition to the problem of deciding how the discourse list should be ordered (a very hard problem in and of itself [8]), the interaction between the local stack and the global discourse list is not clear to us. Consider the example in sentence (12)[8].

(12)	*Tom-i*	*[Mary-ka*	*e*	*hyeppakha-yssta-ko]*	*malha-yssta*
	Tom-NOM	Mary-NOM	EC	threaten-PAST-COMP	say-PAST-DECL

Tom said that Mary threatened e

According to the algorithm given in the previous section, the object argument of *hyeppakhayssta* (*threatened*) should be filled by *Mary*. This, however, is not the case. It is not 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 *e* 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 (12), the empty argument could be filled either by *Tom*, or by something available in the context. The situation in (13) shows a context in which the empty argument in the second sentence would be resolved with *Jerry*, not *Tom*.

(13)

Jerry-ka	*hyeppaktanghaysst a.*
Jerry-NOM	threaten-PASSIVE

Jerry was threatened.

Tom-i	*[Mary-ka*	*e*	*hyeppakha-yssta-ko]*	*malha-yssta.*
Tom-NOM	Mary-NOM	EC	threaten-PAST-COMP	say-PAST-DECL

Tom said that Mary threatened Jerry.

4 Future Work

Work on implementing this empty argument recovery algorithm is being done using a Combinatory Categorial Grammar (CCG) [9] parser written in Prolog. 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 [10]. We are also looking at how centering theory [11, 12] could be used in conjunction with the global ordered list to choose the correct referent for empty arguments that are recovered from NPs outside of the sentence.

5 Conclusion

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 have presented a computational algorithm for locally resolving empty arguments that is based on semantic constraints motivated by the constraints found in scrambling. We have also discussed how this might be used in identifying referents in a discourse-based model.

Notes

[1]The trees in this paper are generated from a Korean grammar [13] implemented in XTAG [14], a grammar development system based on the Feature-based Lexicalized Tree-Adjoining Grammar formalism (FB-LTAG) [15, 16, 17, 18]. Frontier nodes that are not associated with a lexical item can be marked either as **substitution nodes** (↓) or **adjunction nodes** (∗) nodes, indicating the type of operation that they can participate in. Each node in the tree has a set of **feature structures** associated with it (as can be seen in later trees), which can contain additional information about the node or about a lexical item associated with the node. These feature structures can link to other feature structures in the tree, indicating that the nodes must contain the same values (unify) for that particular feature.

[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.

[3]SFX in Figure 2 represents a conjugational suffix form that adjoins onto the verb stem.

[4]Multi-Component TAGs [19], a variation of standard TAGs, is needed to parse sentences that exhibit long-distance scrambling [2].

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

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

[7]There are strong arguments for the subject of the embedded clause to be an instance of PRO-control [20], 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 (see section 3.2), but in this particular case, one would imagine that the last things mentioned would be at the top of the list.

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

References

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

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

[3] Hyun S. Park. Handling of scrambling in Korean using TAGs. Unpublished paper, 1994.

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

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

[6] Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. Providing a unified account of definite noun phrases in discourse. *Proceedings of the 21st Annual Meeting of The Association for Computational Linguistics*, 1983.

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

[8] 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.

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

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

[11] Megumi Kameyama. *Zero Anaphora: the case of Japanese.* PhD thesis, Linguistics Department, Stanford University, 1985.

[12] 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.

[13] Hyun S. Park. Korean grammar using TAGs. Master's Thesis, University of Pennsylvania, In Progress, 1994.

[14] Christy Doran, Dania Egedi, Beth Ann Hockey, B. Srinivas, and Martin Zaidel. Xtag system - a wide coverage grammar for English. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING '94)*, Kyoto, Japan, August 1994.

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

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

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

[18] Aravind K. Joshi, L. Levy, and M. Takahashi. Tree adjunct grammars. *Journal of Computer and System Sciences*, 1975.

[19] David Weir. *Charcterizing Mildly Context-Sensitive Grammar Formalisms.* PhD thesis, University of Pennsylvania, August 1988.

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