PROBLEMS WITH PARSING FREE ORDERED, ELLIPTIC LANGUAGES

Jonathan C. Oh and Steven Graham

Parsing free ordered, elliptic languages raises interesting problems. Korean is a free ordered, verb-final, highly elliptic language, and we will use it for illustration in this paper. We develop some formalism to represent various phenomena that arise in parsing such a language. We will also make suggestions for the implementation of our formalism.

Parsing a free ordered language presents different problems from those found in a language like English. In Korean, components are free ordered within a given clause but only within THAT clause. Components from a higher level clause may not interleave those of a lower level clause, such as a relative clause or a complementation. The free ordered clausal structure of Korean reminds one of the block structures of programming languages such as Pascal. Unfortunately, Korean provides no explicit end markers for its "blocks", allowing multiple parses and resultant ambiguity. We will illustrate this point with the following example:

(1) Kimse_nsan_i e_je_ saosin guduli_l gaje_gassi_bingga
Kim/Mr./subj yesterday buy/honorific shoes/obj take/past/ques
a) 'Did (you or some unspecified person, possibly even Mr. Kim) take the shoes [Mr. Kim bought yesterday]?'
b) 'Did Mr. Kim take the shoes [(you/unspecified) bought] yesterday?'
c) 'Did Mr. Kim take the shoes [(you/unspecified) bought] yesterday]?'
*d) 'Did (you/unspecified) take the shoes [Mr. Kim bought] yesterday?'[1]

[1]The asterisk above indicates the absence of the reading for the given Korean sentence. The brackets are used to show the sentence boundaries. The functional roles are marked in Korean by postpositions.

Language Research, Volume 25, Number 4, December 1989 0254-4474/793-801

Only the first three readings are allowed. The fourth alternative is prohibited by the restriction against interleaving components of different clauses. Thus if Mr. Kim is taken as the subject of the embedded verb 'buy', then the time adverb, 'yesterday', which falls between the subject and the embedded verb, necessarily modifies that verb 'buy' as well. So there is free ordering only among the components of a given clause but never components of a main and subordinate clauses.

Another characteristic that further complicates the parsing process is the high degree of ellipsis common in Korean as well as other languages. Practically anything that can be determined from the context may be omitted from the text. One of the subjects is missing in the above illustration, which has two verbs but only one subject. This relative freedom in omitting components together with the free ordering among the explicit components complicates the parsing process considerably.[2] One cannot assign components to verbs based simply on the types of the verbs and components. Nor can one determine easily which components have been omitted. An unassigned component may be located anywhere within a clause of which it is a part.

The traditional ATN formalism suggests linear structure on the nodes. An extension is made here to accommodate unordered sets of nodes. It also distinguishes those nodes that cannot recur within a given clause, such as NP(subj), NP(obj), and NP(instrumental) from those that may, and often do, recur such as NP(time), NP(place), and adverbs (this latter hereafter referred to as recurrent components). Extending typical ATN notation we mark those components that recur by a Kleene star (indicating zero or more occurrences), while representing the scrambling of component nodes by a pair of braces, the conventional set notation.[3]

We write the following grammar for parsing a subset of Korean using the extensions discussed in the previous paragraph.

[2]A further complication that we will not treat here is the interaction between such contextually controlled deletion and syntactically controlled deletion inside a relative clause. The headnoun of relative clause does not occur inside a relative clause. Notice the ambiguity in the sentence below. The ambiguity arises from two possible deletion schemes allowed due to the lexical choice in the example:

ddalagal ca
follow/rel car
a. the car that will follow
b. the car to follow

[3]This is somewhat like Immediate Dominance relation in GPSG.

S → {NP(subj), NP(obj), NP(instrument),
NP(time)*, NP(place)*, ADVERB*,...} VERB
S → {NP(subj), NP(obj), NP(time)*, NP(place)*,..} VERB
⋮
NP → {relative-clause*, adjective*} NOUN

The main verb of a clause dictates which expansion of 'S' should be used. Interestingly the recurrent components do not appear to participate in subcategorizing verbs. It is possible to give one exhaustive S-expansion rule that contains all nonrecurrent functional components that subcategorize as well as all recurrent ones. Which particular subset of nonrecurrent components is expected in parsing a given sentence may be deduced from the syntactic features in the lexical entry for the verb of that sentence. The verb carries different suffixes to denote different types of sentences, to mark causitivity, passivity, and moods. The type of sentence thus marked dictates different postpositions for the components of the same verb.

(2) gi_gogiga goyan_iege jabhie_ssda
the fish/subj cat/by catch/pass/past/decl
that fish got caught by the cat.
(3) gi_aiga goyan_iege gogili_l jabgehaye_ssda
the child/subj cat/indir fish/obj catch/causative/decl
that child made the cat catch the fish.

In our parser implementation, we take advantage of the verb-final ordering of Korean. We first reverse the input string. Verbs now indicate the beginning of a clause rather than ending one. We can then carry out a morphological analysis of the verb to deduce such extrapropositional information as tense, mood, modality, passivity, and causativity. Lexical analysis also reveals what nonrecurring components might be expected for the verb. A multi-verbal sentence presents a problem of correctly determining clausal boundaries, within which components may be free ordered. The resolution of this problem is found in syntactic, semantic, and pragmatic observations of various sorts. For example, an encounter of two nonrecurrent components of identical function after a verb (in this reversed input) signals the presence of a clausal boundary somewhere between those two components. In the example below, the clause boundary should come somewhere between the two subjects 'Mr Kim' and 'you'.

(4) kimse_nsa_n_i e_je dan sini saosin guduli_l gaje_gassi_bnikka?
Mr. Kim/subj yesterday you/subj buy/hon shoes/obj take/pass/ques
a) Did Mr. Kim take the shoes [that you bought yeaterday]?
b) Did Mr. Kim take the shoes [that you bought] yesterday?

Semantic and pragmatic features also prove useful when filling in missing components or determining which verb a component may be associated with.

(5) sa_ga nalla-gani_nge_si_l ccigi_sie_ssi_laggayo?
bird/subj fly/nominalizn take-photo-honorific/modal
a) Would (unspecified person) have taken a picture of birds flying?
*b) Would the bird have taken a picture of (unspecified) flying?

Reading (a) above presents itself as a strong candidate over the possible but not very likely (b) reading. The preference is suggested here by the type match between 'bird' and 'fly' and by the type mismatch between 'bird' and 'photo-take'. The accuracy of the interpretation thus made is further confirmed by the use of the honorific on the main verb, which is not normally applied to lower creatures.

In such a highly elliptic language, context maintenance is essential. Ultimately it would be necessary to keep track of all events as well as all objects mentioned for future dereferencing. In our parser, we store only those things which are explicitly denoted by noun phrases in what we call the contextual frame. The contextual frame gets augmented with the speaker and the listener to form the superframe. Whenever we detect an empty slot for a non-recurring component in a clause, we fill the slot with elements from the superframe, constrained by semantic and pragmatic restrictions detected while parsing. When more than one element satisfies all the restrictions, the ambiguity persists. The system would need additional information for disambiguation in such a case, just as a human speaker would in the same situation.

We summarize our discussion up to this point in the algorithm below:[4]

1. reverse the input (call it RI)
2. set level n to 1 and call Sentence Parser; we will use two level coun-
;ters, n for verbs and m for
;noun phrases

[4]This algorithm may be coded in a programming language fairly straightforwardly.

```
3. Procedure Sentence Parser
   call Morph-Analyzer for the first element of RI (verb)
    ;;;among other things, gather information as to which components
    ;;;must cooccur (slots that need to be filled in); the type of sentence
    ;;;(passive, causative, imperative, etc); and the function of the sen-
    ;;;tence (relative clause, nominalization, adverbial clause, or main
    ;;;sentence)
  Loop Read Next Word
       if Next word is marked as nonrecurrent NP
        then set level m to 1 and call Parse NP;
             while the returned NP cannot be a component for the current
                level verb
                ;;;this can happen for one of two reasons, one because the
                ;;;current verb cannot take the function of this NP as re-
                ;;;presented by the postposition, or it already has its
                ;;;slot for that function filled
                call Close Verb Structure(n);
                end while;
                Fill in the slots for the function represented by the postposi-
                tion for
                any open verb that allows the function marked by the post-
                position of this NP and include this NP in Contextual
                Frame; if this NP is used for a verb of level n, all verb
                structures get closed for verb of level >n
     else if Next Word is recurrent NP or ADVP
       then either set level m to 1 and call Parse NP or call Parse ADVP;
            assign the result to any open verb structure;
            call Close VerbStructure
     else if Next Word is VERB
       then increment the level counter n by 1;
            recursively call Procedure Sentence Parser for this verb
If RI is empty, then return with the top level (level 1) verb structure
                          ;;;at this point of parse, all expressions are
                          ;;;accounted for
              else go to loop
 end loop
end Sentence Parser
4. Procedure Close Verb Structure(c);
    while n > c do
      Add Hold1, Hold2,...,Holdk where k is the number of components
```

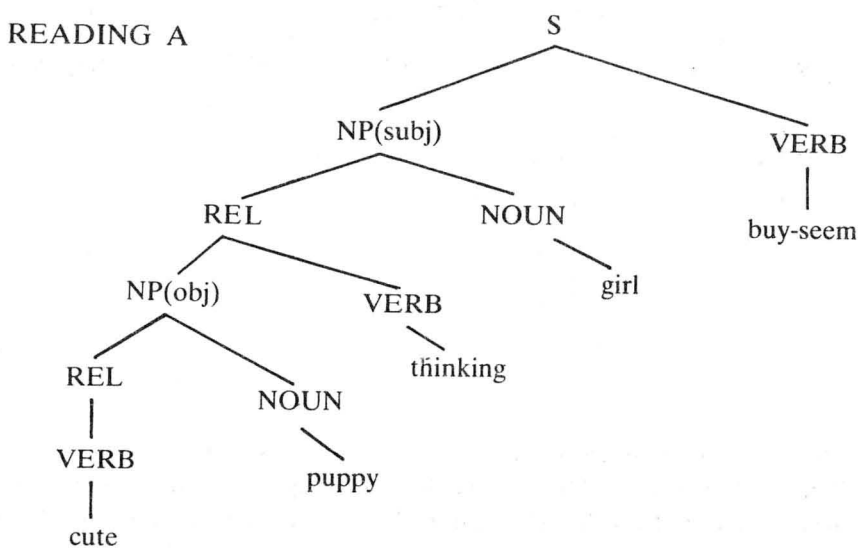
```
        not filled in while parsing for the current level verb (level n)
                                        ;Holdi's are the resisters
                                        ;for empty slots to be
                                        ;filled in
        Decrement n by 1
      end while
    end Close Verb Structure
5. Procedure Parse NP;
        Read Next Word:
        case POS(Next Word) of
            noun: store the parse result and return; we are not concerned
                        ;with details such as compound noun recognition
            verb(relative): while Next Word is a verb (relative) do
                            call Sentence Parser
                            with the parse result fill in the modifier
                              slot for any open NP and adjust the
                              level counter m accordingly.
            adverb: store the parse result and return
            else: store the parse result and return
        end case
    end Parse NP
6. Parse ADVP;
       recognize an adverb or adverbial clause
      end Parse ADVP
```

At this point, we illustrate tha algorithm by parsing the sentence below which is at least 3 ways ambiguous:

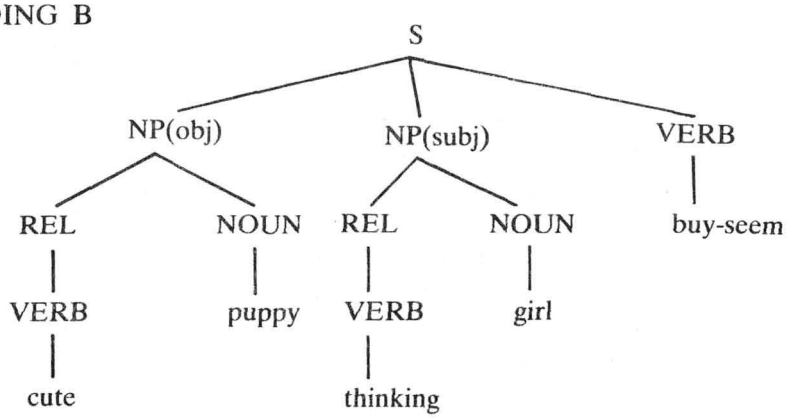
(6) gwiye umi ne mcini n gan ajili l sa n gaghagoissni n sonye ga salge ssgatda
cuteness/subj overflow/rel puppy/obj think/prog/rel girl/subj buy/presum/decl
a. It seems that the girl who is thinking of the cute puppy will buy-it.
b. It seems that the girl who is in deep thought will buy the cute puppy.
c. It seems that the cute girl who is thinking of the puppy will buy-it.

According to our algorithm, the input sentence is reversed, after which we call Sentence Parser. Morph-Analyzer after analyzing the verb

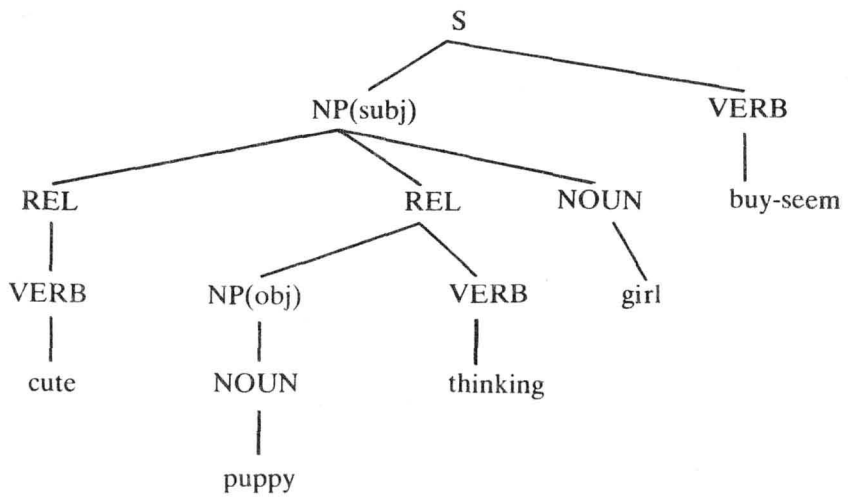
'salge_ssgatda' finds out among other things that we are dealing with a main sentence here and that the verb is a transitive verb. The two required nonrecurrent components, therefore, are NP(subj) and NP(obj). The next word is a noun with the subject marker. We call Parse NP. In Parse NP, we encounter the next word which indicates a relative clause. We recursively call Parse Sentence from Parse NP. Now we are looking at the level 2 verb (n=2), sa_n_gaghagoissni_n 'thinking of'. The next word at this point is an object case NP gan_ajili-_l 'puppy(obj)'. We call Parse NP with m=2. Now we nondeterministically assign this object NP, to either the level 1 verb 'sell' or the level 2 verb 'thinking of'. The former gives reading b and the latter reading a or c depending on the rest of the parse. If we choose the former, the level 2 verb structure gets closed. If not, the parse continues with the reading of the last word 'that is cute.' Since this word is a verb(relative), we call Parse Sentence with n=c. Level 3 sentence parse succeeds unambiguously since the verb happens to be the last word in RI. We have another choice in assigning this relative clause since m=2. If we assign this relative clause as a modifier to the level 1 noun (girl), the level 2 NP gets closed giving 'girl thinking of a puppy' as the subject of the main verb (reading c). Otherwise, we get reading b.



READING B



READING C



The source of nondeterminism is the keyword 'any' in Sentence Parser and Parse NP algorithm. The familiar backtracking or parallel parsing strategies may be adopted for disambiguation. But we recommend the use of Seo and Simmons' parse graphs [Seo 89] for the parse results. Disambiguation and filling of hold registers will occur at a later stage using the contents of Contextual Frame and other discourse information.

References

Cullingford, R. E. (1986) *Natural Language Processing: a Knowledge-Engineering Approach,* Rowman & Littlefield.

King, M. ed. (1983) *Parsing Natural Language,* Academic Press.

Seo, J. and R. F. Simmons (1989) 'Syntactic Graphs: A Representation for the Union of All Ambiguous Parse Tress,' *Computational Linguistics* 15 : 1.

Tomita, M. (1986) *Efficient Parsing for Natural Language,* Kluwer Academic Publishers.

Computer Science
University of Missouri at K. C.
Kansas City, MO 64110
U.S.A.