Parsing with Discontinuous Constituents

Mark Johnson
Center for the Study of Language and Information
and
Department of Linguistics, Stanford University.

Abstract

By generalizing the notion of *location* of a constituent to allow discontinuous loctaions, one can describe the discontinuous constituents of non-configurational languages. These discontinuous constituents can be described by a variant of definite clause grammars, and these grammars can be used in conjunction with a proof procedure to create a parser for non-configurational languages.

1. Introduction

In this paper I discuss the problem of describing and computationally processing the discontinuous constituents of non-configurational languages. In these languages the grammatical function that an argument plays in the clause or the sentence is not determined by its position or configuration in the sentence, as it is in configurational languages like English, but rather by some kind of morphological marking on the argument or on the verb. Word order in non-configurational languages is often extremely free: it has been claimed that if some string of words S is grammatical in one of these languages, then the string S′ formed by any arbitrary permutation of the words in S is also grammatical. Most attempts to describe this word order freedom take mechanisms designed to handle fairly rigid word order systems and modify them in order to account for the greater word order freedom of non-configurational languages. Although it is doubtful whether any natural language ever exhibits such total scrambling, it is interesting to investigate the computational and linguistic implications of systems that allow a high degree of word order freedom. So the approach here is the opposite to the usual one: I start with a system which, unconstrained, allows for unrestricted permutation of the words of a sentence, and capture any word order regularities the language may have by adding restrictions to the system. The extremely free word order of non-configurational languages is described by allowing constituents to have discontinuous locations. To demonstrate that it is possible to parse with such discontinuous constituents, I show how they can be incorporated into a variant of definite clause grammars, and that these grammars can be used in conjunction with a proof procedure, such as Earley deduction, to construct a parser, as shown in Pereira and Warren (1983).

This paper is organized as follows: section 2 contains an informal introduction to Definite Clause Grammars and discusses how they can be used in parsing, section 3 gives a brief description of some of the grammatical features of one non-configurational language, Guugu Yimidhirr, section 4 presents a definite clause fragment for this language, and shows how this can be used for parsing. Section 5 notes that the use of discontinuous constituents is not limited to definite clause grammars, but they could be incorporated into such disparate formalisms as GPSG, LFG or GB. Section 6 discusses whether a unified account of parsing both configurational and non-configurational languages can be given, and section 7 compares the notion of discontinuous constituents with other approaches to free word order.

2. Definite Clause Grammars and Parsing

In this section I show how to represent both an utterance and a context free grammar (CFG) so that the locations of constituents are explicitly represented in the grammar formalism. Given this, it will be easy to generalize the notion of location so that it can describe the discontinuous constituents of non-configurational languages. The formalism I use here is the Definite Clause Grammar formalism described in Clocksin and Mellish (1984). To familiarize the reader with the DCG notation, I discuss a fragment for English in this section. In fact, the DCG representation is even more general than is brought out here: as Pereira and Warren (1983) demonstrated, one can view parsing algorithms highly specialized proof procedures, and the process of parsing as logical inferencing on the representation of the utterance, with the grammar functioning as the axioms of the logical system.

Given a context free grammar, as in (1), the parsing problem is to determine whether a particular utterance, such as the one in (2), is an S with respect to it.

(1) $\mathrm{S} \rightarrow \mathrm{NP}\ \mathrm{VP}$
$\mathrm{VP} \rightarrow \mathrm{V}\ \mathrm{NP}$
$\mathrm{NP} \rightarrow \mathrm{Det}\ \mathrm{N}$
$\mathrm{Det} \rightarrow \underset{[+\mathrm{Gen}]}{\mathrm{NP}}$

(2) ${}_0$ the ${}_1$ boy's ${}_2$ father ${}_3$ hit ${}_4$ the ${}_5$ dog ${}_6$

The subscripts in (2) serve to *locate* the lexical items: they indicate that, for instance, the utterance of the word *dog* began at time t_5 and ended at time t_6. That is, the *location* of the utterance "dog" in example (2) was the interval $[t_5,t_6]$. I interpret the subscripts as the points in time that segment the utterance into individual words or morphemes. Note that they perform the same function as the *vertices* of a standard chart parsing system.

Parsing (2) is the same as searching for an S node that dominates the entire string, ie. whose location is $[0,6]$. By looking at the rules in (1), we see that an S node is composed of an NP and a VP node. The interpretation conventions associated with phrase structure rules like those in (1) tell us this, and also tell us that the location of the S is the concatenation of the location of the NP and the VP. That is, the existence of an S node located at $[0,6]$ would be implied by the existence of an NP node located at interval $[0,x]$ (x a variable) and a VP node located at $[x,6]$.

The relationship between the mother constituent's location and those of its daughters is made explicit in the definite clause grammar, shown in (3), that corresponds to the CFG (1). The utterance (1) (after lexical analysis) would be represented as in (4). Those familiar with Prolog should note that I have reversed the usual orthographic convention by writing variables with a lower case intial letter (they are also italicized), while constants begin with an upper case letter.

(3) $\mathrm{S}(x,z) \leftarrow \mathrm{NP}(x,y,\emptyset)\ \&\ \mathrm{VP}(y,z).$
$\mathrm{VP}(x,z) \leftarrow \mathrm{V}(x,y)\ \&\mathrm{NP}(y,z,\emptyset).$
$\mathrm{NP}(x,z,case) \leftarrow \mathrm{Det}(x,y)\ \&\ \mathrm{N}(y,z,case).$
$\mathrm{Det}(x,y) \leftarrow \mathrm{NP}(x,y,\mathrm{Gen})$

(4) Det(0,1).
N(1,2,Gen).
N(2,3,∅).
V(3,4).
Det(4,5).
N(5,6,∅).

(3) contains four definite clauses; each corresponds to one phrase structure rule in the CFG (1). The major difference for us between (1) and (3) is that in (3) the locations of the constituents (ie. the endpoints) are explicit in the formalism: they are arguments of the constituents, the first argument being the beginning time of the constituent's location, the second argument its ending time. Also note the way that syntactic features are treated in this system: the difference between genitive and non-genitive case marked nouns is indicated by a third argument in both the N and NP constituents. Genitive nouns and noun phrases have the value Gen for their third argument, while non-genitive NP have the value ∅, and the rule that expands NP explicitly passes the case of the mother to the head daughter[1].

We can use (3) and (4) to make inferences about the existence of constituents in utterance (2). For example, using the rule that expands NP in (3) together with the first two facts of (4), we can infer the existence of constituent NP(0,2,Gen).

The simplest approach to parsing is probably to use (1) in a top-down fashion, and start by searching for an S with location [0,6]; that is, search for goal S(0,6). This method, top down recursive descent, is the method the programming language Prolog uses to perform deduction on definite clauses systems, so Prolog can easily be used to make very efficient top-down parsers.

Unfortunately, despite their intuitive simplicity, top down recursive descent parsers have certain properties that make them less than optimal for handling natural language phenomena. Unless the grammar the parser is using is very specially crafted, these parsers tend to go into infinite loops. For example, the rule that expands NP into Det and N in (3) above would be used by a top-down parser to create a Det subgoal from an NP goal. But Det itself can be expanded as a genitive NP, so the parser would create another NP subgoal from the Det subgoal, and so on infinitely.

The problem is that the parser is searching for the same NP many times over: what is needed is a strategy that reduces multiple searches for the same item to a single search, and arranges to share its results. Earley deduction, based on the Earley parsing algorithm, is capable of doing this. For reasons of time, I won't go into details of Earley Deduction (see Pereira and Warren (1983) for details); I will simply note here that using Earley Deduction on the definite clause grammar in (3) results in behaviour that corresponds exactly to the way an Earley chart parser would parse (1).

3. Non-configurational Languages

In this section I identify some of the properties of non-configurational languages. Since this is a paper on discontinuous constituents, I focus on word order properties, as exemplified in the non-configurational language Guugu Yimidhirr. The treatment here is necessarily superficial: I have completely ignored many complex phonological, inflectional and syntactic processes that a complete grammar would have to deal with.

A non-configurational language differs from configurational languages like English in that morphological form (eg. affixes), rather than position (ie. configuration), indicates which words are syntactically connected to each other. In English the grammatical, and hence semantic, relationship between *boy*, *father* and *dog* in (5) are indicated in surface form by their positions, and changing these positions changes these relationships, and hence the meaning, as in (6).

(5) The boy's father hit the dog

(6) The father's dog hit the boy

In Guugu Yimidhirr, an Australian language spoken in north-east Queensland, the relationships in (7)[2] are indicated by the affixes on the various nouns, and to change the relationships, one would have to change the affixes.

(7)

Yarraga-aga-mu-n	gudaa	gunda-y	biiba-ngun
boy-GEN-mu-ERG	dog+ABS	hit-PAST	father-ERG

'The boy's father hit the dog'

The idea, then, is that in these languages morphological form plays the same role that word order does in a configurational language like English. One might suspect that word order would be rather irrelevant in a non-configurational language, and infact

> Guugu Yimidhirr speakers remark that their language, unlike English, can be spoken 'back to front': that is, it is possible to scramble words and still produce a grammatical utterance (Haviland 1979, p 26.)

Interestingly, in some Guugu Yimidhirr constructions it appears that information about grammatical relations can be obtain either through word order or morphology: in the possessive construction

> When a complex NP carries case inflection, each element (in this case, both possession and possessive expression) may bear case inflection - and both *must* be inflected for case if they are not contiguous - but frequently the 'head noun' (the possession) [directly MJ] precedes the possessive expression, and only the latter has explicit case inflection (Haviland 1979,p.56)

Thus in (8), *biiba* 'father' shows up without an ergative suffix because it is immediately to the left of the NP that possesses it (ie. possession is indicated by position).

(8)

Biiba yarraga-aga-mu-n	gudaa	gunda-y	
father boy-GEN-mu-ERG	dog+ABS	hit-PAST	

'The boy's father hit the dog'

While ultimate judgement will have to await a full analysis of these constructions, it does seem as if word order and morphological form do supply the same sort of information.

In the sections that follow, I will show how a variant of definite clause grammar can be used to describe the examples given above, and how this grammar can be used in conjunction with a proof procedure to construct a parser.

4. Representing Discontinuous Constituents

I propose to represent discontinuous constituents rather directly, in terms of a syntactic category and a discontinuous location in the utterance. For example, I represent the location of the discontinous constituent in (7), *Yarraga-aga-mu-n* ... *biiba-ngun* 'boy's father' as a *set* of continuous locations, as in (9).

(9) {[0,1],[3,4]}

Alternatively, one could represent discontinuous locations in terms of a 'bit-pattern', as in (10), where a '1' indicates that the constituent occupies this position.

(10) [1 0 0 1]

While the descriptive power of both representations is the same, I will use the representation of (9) because it is somewhat

[1] Of course, there is nothing special about these two values: any two distinct values would have done.

[2] All examples are from Haviland (1979). The constructions shown here are used to indicate *alienable* possession (which includes kinship relationships).

easier to state configurational notions in it. For example, the requirement that a constituent be contiguous can be expressed by requiring its location set to have no more than a single interval member.

To represent the morphological form of NP constituents I use two argument positions, rather than the single argument position used in the DCG in (3). The first takes as values either Erg , Abs or ∅, and the second either Gen or ∅. Thus our discontinuous NP has three argument positions in total, and would be represented as (11).

(11) NP([[0,1],[3,4]],Erg,∅).

In (11), the first argument position identifies the constituent's location, while the next two are the two morphological form arguments discussed immediately above. The grammar rules must tell us under what conditions we can infer the existence of a constituent like (11). The morphological form features seem to pose no particular problem: they can be handled in a similiar way to the genitive feature in the mini-DCG for English in (3) (although a full account would have to deal with the dual ergative/absolutive and nominative/accusative systems that Guugu Yimidhirr possesses). But the DCG rule format must be extended to allow for discontinuous locations of constituents, like (11).

In the rules in (3), the end-points of the mother's location are explicitly constructed from the end-points of the daughter's locations. In general, the realtionship between the mother's location and that of its daughters can be represented in terms of a predicate that holds between them. In the DCG rules for Guugu Yimidhirr, (12) to (14), the relationship between the mother's location and those of its daughters is represented by the predicate *combines*.

The definition of *combines* is as follows: combines(l,l_1,l_2) is true if and only if l is equal to the (bit-wise) union of l_1 and l_2, and the (bit-wise) intersection of l_1 and l_2 is null (ie. l_1 and l_2 must be non-overlapping locations).

(12) S(l) ← V(l_1) & NP(l_2,Erg,∅) & NP(l_3,Abs,∅)
& combines(l,l_1,l_2,l_3).
(13) NP(l,*case*,∅) ← N(l_1,*case*,∅) & NP(l_2,*case*,Gen)
& combines(l,l_1,l_2).
(14) NP(l,$case_1$,$case_2$) ← N(l,$case_1$,$case_2$)

Following Hale (1983), I have not posited a VP node in this grammar, although it would have trivial to do so. To account for the 'configurational' possessive shown in (8), I add the additional clause in (15) to the grammar.

(15) NP([[x,z]],*case*,∅) ← NP([[x,y]],∅,∅) & N([[y,z]],*case*,Gen)

Given this definite clause grammar and a proof procedure such as Earley Deduction, it is quite straight-forward to construct a parser. In (16) I show how one can parse (7) using the above grammar and the Earley Deduction proof procedure. The Prolog predicate 'p' starts the parser. First the lexical analyser adds lexical items to the state (the working store of the deduction procedure), when this is finished the deduction procedure uses the DCG rules above to make inferences about the utterance. The answer '**yes' given by the parser indicates that it was able to find an S that spans the entire utterance. The command 'print_state' prints the state of the deduction system; since new inferences are always added to the bottom of the state, it provides a chonological records of the deductions made by the system. The state is printed in Prolog notation: variables are written as '_1', '_2', etc., and the implication symbol ← as ':-'.

(16)
```
%prolog earley aux nscr

UNSW - PROLOG

: p([yarragaagamun,gudaa,gunday,biibangun])?
Word yarragaagamun is a n([[0, 1]], erg, gen)
Word gudaa is a n([[1, 2]], abs, o)
Word gunday is a v([[2, 3]])
Word biibangun is a n([[3, 4]], erg, o)
** yes
: print_state !
n([[0, 1]], erg, gen)
n([[1, 2]], abs, o)
v([[2, 3]])
n([[3, 4]], erg, o)
s([[0, 4]]) :- v(_1) , np(_2, erg, o) , np(_3, abs, o) ,
        combines([[0, 4]], _1, _2, _3)
s([[0, 4]]) :- np(_1, erg, o) , np(_2, abs, o) ,
        combines([[0, 4]], [[2, 3]], _1, _2)
np(_1, erg, o) :- n(_2, erg, o) , np(_3, erg, gen) ,
        combines(_1, _2, _3)
np(_1, erg, o) :- np(_2, erg, gen) , combines(_1, [[3, 4]], _2)
np(_1, erg, gen) :- n(_1, erg, gen)
np([[0, 1]], erg, gen)
np(_1, erg, o) :- combines(_1, [[3, 4]], [[0, 1]])
combines([[0, 1], [3, 4]], [[3, 4]], [[0, 1]])
np([[0, 1], [3, 4]], erg, o)
s([[0, 4]]) :- np(_1, abs, o) ,
        combines([[0, 4]], [[2, 3]], [[0, 1], [3, 4]], _1)
np(_1, abs, o) :- n(_2, abs, o) , np(_3, abs, gen) ,
        combines(_1, _2, _3)
np(_1, abs, o) :- np(_2, abs, gen) , combines(_1, [[1, 2]], _2)
np(_1, abs, gen) :- n(_1, abs, gen)
np(_1, abs, o) :- n(_1, abs, o)

np([[1, 2]], abs, o)
s([[0, 4]]) :- combines([[0, 4]], [[2, 3]], [[0, 1], [3, 4]], [[1, 2]])
combines([[0, 4]], [[2, 3]], [[0, 1], [3, 4]], [[1, 2]])
s([[0, 4]])
np([[_1, _2]], abs, o) :- np([[_1, _3]], o, o) , n([[_3, _2]], abs, gen)
np([[_1, _2]], o, o) :- n(_3, o, o) , np(_4, o, gen) ,
        combines([[_1, _2]], _3, _4)
np([[_1, _2]], o, o) :- n([[_1, _2]], o, o)
np([[_1, _2]], o, o) :- np([[_1, _3]], o, o) , n([[_3, _2]], o, gen)
np(_1, erg, o) :- n(_1, erg, o)
np([[3, 4]], erg, o)
s([[0, 4]]) :- np(_1, abs, o) , combines([[0, 4]], [[2, 3]], [[3, 4]], _1)
s([[0, 4]]) :- combines([[0, 4]], [[2, 3]], [[3, 4]], [[1, 2]])
np([[_1, _2]], erg, o) :- np([[_1, _3]], o, o) , n([[_3, _2]], erg, gen)
: ^D
```

5. Using Discontinuous Constituents in Grammars

Although the previous section introduced discontinuous constituents in terms of definite clause grammar, there is no reason we could not invent a notation that abbreviates or implies the 'combines' relationship between mother and daughters, just as the CFG in (1) "implies" the mother-daughter location relationships made explicit in the DCG (3). For instance, we could choose to interpret a rule like (17) as implying the 'combines' relationship between the mother and its daughters.

(17) $A \rightarrow B\ ;\ C\ ;\ D$

Then the DCG grammar presented in the last section could be written in the GPSG like notation of (18). Note that the third rule is a standard phrase structure rule: it expresses the 'configurational' possessive shown in (8).

(18) $$S \rightarrow \underset{[\text{CASE Erg}]}{\text{NP}} \; ; \; V \; ; \; \underset{[\text{CASE Abs}]}{\text{NP}}$$

$$\underset{[\text{CASE } \alpha]}{\text{NP}} \rightarrow \left(\begin{bmatrix} \text{NP} \\ \text{Gen+} \\ \text{CASE } \alpha \end{bmatrix} \right) ; \underset{[\text{CASE } \alpha]}{\text{N}}$$

$$\underset{[\text{CASE } \alpha]}{\text{NP}} \rightarrow \begin{bmatrix} \text{NP} \\ \text{Gen+} \\ \text{CASE } \alpha \end{bmatrix} .$$

It is easy to show that grammars based on the 'combines' predicate lie outside the class of context free languages: the strings the grammar (19) accepts are the permutations of $a^n b^n c^n$; thus this grammar does not have weakly equivalent CFG.

(19) $S \rightarrow a \; ; \; b \; ; \; c \; ; \; (S)$

While it would be interesting to investigate other properties of the 'combines' predicate, I suspect that it is not optimal for describing linguistic systems in general, including non-configurational languages. It is difficult to state word order requirements that refer to a particular constituent position in the utterance. For instance, the only word order requirement in Warlpiri, another non-configurational language, is that the auxilary element must follow exactly one syntactic constituent, and this would be difficult to state in a system with only the predicate 'combines', although it would be easy to write a special DCG predicate which forces this behaviour.

Rather, I suspect it would be more profitable to investigate other predicates on constituent locations besides 'combines' to see what implications they have. In particular, the wrapping operations of Pollard (1984) would seem to be excellent candidates for such research.

Finally, I note that the discontinuous constituent analysis described here is by no means incompatible with standard theories of grammar. As I noted before, the rules in (18) look very much like GPSG rules, and with a little work much of the machinery of GSPG could be grafted on to such a formalism. Similiarly, the CFG part of LFG, the C-structure, could be enriched to allow discontinuous constituents if one wished. And introducing some version of discontinuous constituents to GB could make the mysterious "mapping" between P-structure and L-structure that Hale (1983) talks about a little less perplexing.

My own feeling is that the approach that would bring the most immediate results would be to adopt some of the "head driven" aspects of Pollard's (1984) Head Grammars. In his conception, heads contain as lexical information a list of the items they subcategorize for. This strongly suggests that one should parse according to a "head first" strategy: when one parses a sentence, one looks for its verb first, and then, based on the lexical form of the verb, one looks for the other arguments in the clause. Not only would such an approach be easy to implement in a DCG framework, but given the empirical fact that the nature of argument NPs in a clause is strongly determined by that clause's verb, it seems a very reasonable thing to do.

6. Implementing the Parser

In their 1983 paper, Pereira and Warren point out several problems involved in implementing the Earley proof procedure, and proposed ways of circumventing or minimizing these problems. In this section I only consider the specialized case of Earley Deduction working with clauses that correspond to grammars of either the continuous or discontinuous constituent type, rather than the general case of performing deduction on an arbitrary set of clauses.

Considering first the case of Earley Deduction applying to a set of clauses like (3) that correspond to a CFG, a sensible thing to do would be to index the derived clauses (ie. the intermediate results) on the left edge of their location. Because Earley Deduction on such a set of clauses always proceeds in exactly the same manner as Earley chart parsing, namely strictly left to right within a constituent, the position of the left edge of any constituent being searched for is always determined by the ending location of the constituent immediately preceeding it in the derivation. That is, the proof procedure is always searching for constituents with hard, ie. non-variable, left edges. I have no empirical data on this point, but the reduction in the number of clauses that need to be checked because of this indexing could be quite important. Note that the vertices in a chart act essentially as indices to edges in the manner described.

Unfortunately, indexing on the left edge in system working with discontinuous constituents in the manner suggested above would not be very useful, since the inferencing does not proceed in a left to right fashion. Rather, if the suggestions at the end of the last section are heeded, the parser proceeds in a "head first" fashion, looking first for the head of a constituent and then for its complements, the nature and number of which are partially determined by information available from the head. In such a strategy, it would seem reasonable to index clauses not on their location, but on morphological or categorial features, such as category, case, etc., since these are the features they will be identified by when they are searched for.

It seems then that the optimal data structure for one type of constituent is not optimal for the other. The question then arises whether there is a unified parsing strategy for both configurational and non-configurational languages. Languages with contiguous constituents could be parsed with a head first strategy, but I suspect that this would prove less efficient than a strategy that indexed on left edge position. Locations have the useful property that their number grows as the size of the sentence (and hence the number of constituents) increases, thus giving more indexing resolution where it is needed, namely in longer sentences. But of course, one could always index on *both* morphological category and utterance location...

7. Comparison with other Frameworks

In this section I compare the discontinuous location approach I have developed above to some other approaches to free word order: the ID/LP rule format of GPSG, and the non-configurational encoding of LFG. I have omitted a discussion of the scrambling and raising rules of Standard Theory and their counterparts in current GB theory because their properties depend strongly on properties of the grammatical system as a whole (such as a universal theory of "landing sites", etc.): which (as far as I know) have not been given in sufficiently specific form to enable a comparison.

The ID/LP rule format (Gazdar et al. 1985) can be regarded as a factoring of "normal" context free rules[3] into two components, one expressing *immediate domination* relationships, the other the *linear precedence* relationships that hold between daughter constituents. For example, the ID rule in (20) and the LP rule in (21) express the same mother-daughter relationships as the rules in (22).

(20) $S \rightarrow_{ID} \{V, NP, NP, S'\}$

(21) $V < S'$

(22)
$S \rightarrow V\ NP\ NP\ S'$
$S \rightarrow V\ NP\ S'\ NP$
$S \rightarrow V\ S'\ NP\ NP$
$S \rightarrow NP\ V\ NP\ S'$
$S \rightarrow NP\ V\ S'\ NP$
$S \rightarrow NP\ NP\ V\ S'$

Because a grammar in ID/LP format always has a strongly equivalent context free grammar, only context free languages can be generated by these grammars. Since it is possible to write grammars

[3] In Gazdar et al. (1985) the system is more complicated than this, since the ID/LP component interacts with the feature instantiation principles and other components of the grammar.

for non-context-free languages using discontinuous constituents (as shown above), it is clear that ID/LP format is less powerful than the discontinuous constituent analysis proposed here. In particular, ID/LP allows only reordering of the daughters of a constitiuent relative to the other daughters: it does not allow a constituent to be "scattered" accross the sentence in the way a discontinuous constituent analysis allows. Thus an ID/LP grammar of Guugu Yimidhirr could not analyse sentence (7) in the same way we did here. In fact, if we added the requirement that all locations be continuous (ie. that the location sets contain at most one member) to the DCG rules using the 'combines' predicate, the word order freedom allowed would be the same as that allowed by an ID rule without any LP restrictions. I don't claim that it is impossible to write a GPSG grammar for a language like Guugu Yimidhirr on the basis of the formalism's not allowing discontinuous constituents: on closer investigation it might turn out that the "discontinuities" could be described by some set of medium or long distance dependencies.

In LFG the nature of the mapping between c-structure and f-structure enables it to achieve many of the effects of discontinuous constituents, even though the phrase structure component (the c-structure) does not allow discontinuous constituents as such. In particular, the information represented in one component of the f-structure may come from several different c-structure constituents located throughout the sentence. For example, in their analysis of the cross serial dependencies in Dutch, Bresnan, Kaplan, Peters and Zaenen (1982) propose that the PRED feature of the VCOMP component of the f-structure is set by a verb located down one branch of the c-structure tree, while the OBJ feature of that component is set by an NP located on another branch of the c-structure tree. Thus in LFG one would not claim that there was a discontinuous NP in (7), but rather that both the ergative NP and the genitive marked ergative NP were contributing information to the same component of the f-structure.

In the ***non-configurational encoding*** of Bresnan (1982, p.297), the c-structure is relatively impoverished, and the morphology on the lexical items identifies the component of the f-structure they supply information to. For example, the c-structure in (23) together with the lexical items in (24) give sentence (7) the f-structure (25).

(23) $$S \rightarrow \left\{ \begin{matrix} \mathrm{NP} \end{matrix} , \begin{matrix} \mathrm{V} \\ \uparrow=\downarrow \end{matrix} \right\}^{*}$$

(24)

yarraga-aga-mu-n	NP	$(\uparrow \mathrm{SUBJ\ POSS})=\downarrow$ $(\downarrow \mathrm{CASE})=\mathrm{Erg}$ $(\downarrow \mathrm{Gen})=+$
gudaa	NP	$(\uparrow \mathrm{OBJ})=\downarrow$ $(\downarrow \mathrm{CASE})=\mathrm{Abs}$
gunda-y	V	$(\uparrow \mathrm{PRED})=\mathit{hit}((\uparrow \mathrm{SUBJ}),(\uparrow \mathrm{OBJ}))$
biiba-ngun	NP	$(\uparrow \mathrm{SUBJ})=\downarrow$ $(\downarrow \mathrm{CASE})=\mathrm{Erg}$

(25)

$$\left[\begin{array}{ll} \mathrm{SUBJ} = & \left[\begin{array}{ll} \mathrm{POSS} = & \left[\begin{array}{l} \mathrm{CASE} = \mathrm{Erg} \\ \mathrm{Gen} = + \\ \mathrm{PRED} = \mathit{boy} \end{array} \right] \\ \mathrm{CASE} = \mathrm{Erg} & \\ \mathrm{PRED} = \mathit{father} & \end{array} \right] \\ \mathrm{OBJ} = & \left[\begin{array}{l} \mathrm{CASE} = \mathrm{Abs} \\ \mathrm{PRED} = \mathit{dog} \end{array} \right] \\ \mathrm{PRED} = \mathit{hit}(\;,\;) & \end{array} \right]$$

LFG is capable of describing the "discontinuity" of (7) without using discontinuous constituents. There is, however, a subtle difference in the amount of "discontinuity" allowed by the LFG and the discontinuous constituent analyses. As I remarked at the beginning of the paper, the discontinuous constituent approach allows grammars that accept total scrambling of the lexical items: if a string S is accepted, then so is any permutation of S. In particular, the discontinuous constituent approach allows unrestricted scrambling of elements out of embedded clauses and stacked NPs, which the LFG non-configurational encoding analysis cannot. This is because the position in the sentence's f-structure that any lexical item occupies is determined solely by the f-equation annotations attached to that lexical item, since the only equations in the c-structure are of the form $\uparrow=\downarrow$, and these create no new components in the f-structure for the clause to embed the f-structures from lexical items into.

Suppose, for example, Guugu Yimidhirr allowed stacked NP possessors, in the same way that English allows them in constructions like *my mother's father's brother*, except that, because the language is non-configurational, the lexical elements could be scattered throughout the entire sentence. The LFG analysis would run into problems here, because there would be a potentially infinite number of positions in the f-structure where the possessor could be located: implying that there are an infinite number of lexical entries for each possessive NP.

Guugu Yimidhirr does not exhibit such stacked possessives. Rather, the possessor of the possessor is indicated by a dative construction and so the LFG analysis is supported here. None the less, a similiar argument shows that embedded clausal f-structure components such as adjuncts or VCOMP must have corresponding c-structure nodes so that the lexical items in these clauses can be attached sufficiently "far down" in the f-structure for the entire sentence. (Another possibility, which I won't explore here, would be to allow f-equation annotations to include *regular expressions* over items like VCOMP). Still, it would be interesting to investigate further the restrictions on scrambling that follow from the non-configurational encoding analysis and the basic principles of LFG. For instance, the *offline parsability* property (Pereira and Warren 1983) that is required to assure decidablity in LFG (Bresnan and Kaplan 1982) essentially prohibits scrambling of single lexical elements from doubly embedded clauses, because such scrambling would entail one S node exhaustively dominating another. But these distinctions are quite subtle, and, unfortunately, our knowledge of non-configurational languages is insufficient to determine whether the scrambling they exhibit is within the limits allowed by non-configurational encoding.

8. Conclusion

Hale (1983) begins his paper by listing three properties that have come to be associated with the typological label 'non-configurational', namely (i) free word order, (ii) the use of syntactically discontinuous constituents and (iii) the extensive use of null anaphora. In this paper I have shown that the first two properties follow from a system that allows constituents that have discontinuous constituents and that captures the mother daughter location relationships using a predicate like 'combines'.

It is still far too early to tell whether this approach really is the most appropriate way to deal with discontinuous constituents: it may be that for a grammar of reasonable size some other technique, such as the non-configurational encoding of LFG, will be superior on linguistic and computational grounds.

9. Bibliography

J. Bresnan (1982), "Control and Complementation," in *The Mental Representation of Grammatical Relations*, J. Bresnan, ed., pp. 173-281, MIT Press, Cambridge, Mass.

J. Bresnan and R. Kaplan (1982), "Lexical Functional Grammar: A Formal System for Grammatical Representation," in *The Mental Representation of Grammatical Relations*, J. Bresnan, ed., pp. 173-281, MIT Press, Cambridge, Mass.

J. Bresnan, R. Kaplan, S. Peters and A. Zaenen (1982), *Cross-Serial Dependencies in Dutch*, Linguistic Inquiry, 11.4, pp. 613-635.

G. Gazdar, E. Klein, G. Pullum and I. Sag, (1985) *Generalized Phrase Structure Grammar*, Havard University Press, Cambridge, Mass.

K. Hale (1983), "Warlpiri and the Grammar of Non-configurational Languages", *Natural Language and Linguistic Theory*, 1.1, pp. 5-49.

J. Haviland (1979), "Guugu Yimidhirr", in *Handbook of Australian Languages*, R. Dixon and B. Blake, eds., Benjamins, Amsterdam.

F.C.N. Pereira and D.H.D. Warren (1983), "Parsing as Deduction", *Proc. of the 21st Annual Meeting of the ACL*, pp. 137-143, Association for Computational Linguistics.

C. Pollard (1984), *Generalized Phrase Structure Grammars, Head Grammars, and Natural Language*, unpublished thesis, Stanford University.