Formal tools for describing and processing discontinuous constituency structure[1]

Harry Bunt
Institute for Language Technology and Artificial Intelligence ITK
P.O.Box 90153, 5000 LE Tilburg, The Netherlands

Abstract

This contribution is concerned with the development of formal tools for the description and processing of discontinuous constituency structures. It is argued first that the traditional tree representations of constituent structure are too limited, and that these representations in general should be allowed to have crossing branches. A precise mathematical definition of *discontinuous trees* is provided, plus a linear precedence relation for expressing a kind of concatenation operation on these structures.

The formal concepts introduced here are theory-neutral and can be applied in a wide variety of grammatical frameworks and their computer implementations. Their use in phrase-structure grammars is explained in some detail, developing a grammatical formalism called *Discontinuous Phrase Structure Grammar* (DPSG); their possible use in Categorial Grammar is considered very briefly.

The paper concludes with a section on computational issues, in particular in relation to the computer implementation of DPSG for parsing and generation.

1 Introduction

This paper takes the view that discontinuity in constituent structure is nothing to be afraid of, and gives substance to this view by developing formal tools that can be used both in the description of discontinuous constituency in a wide range of grammatical formalisms, and in the formulation of efficient algorithms for parsing and generating sentences with discontinuities.

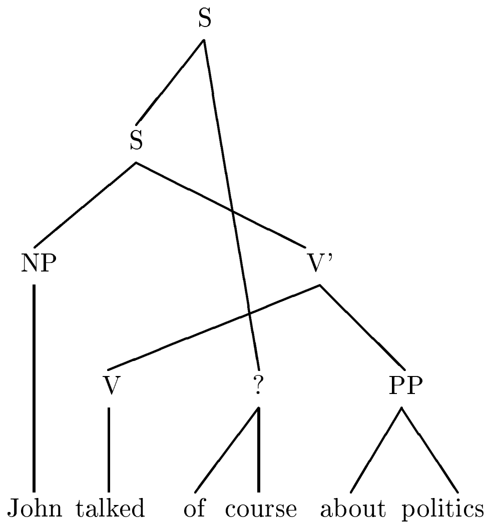
One way of expressing the basic idea underlying this paper is that constituents can fruitfully be regarded not as *sequences* of words, but as partially

[1] *H. Bunt and A. van Horck (eds.), Discontinuous Constituency, 63–85.*

ordered *sets* of words[2] with certain precedence relations amongst themselves and with surrounding words. Central in the elaboration of this idea is the notion of a *discontinuous tree*, which is a tree-like structure with crossing branches describing discontinuous constituency.

The idea of allowing discontinuous constituent structures is not new; for example, McCawley (1982) has suggested to analyse the sentence *John talked of course about politics* as having the constituent structure displayed in (1). We will return to his proposal below.

(1)



When it is not possible to analyse a given sentence into continuous NPs, VPs, APs, etc., we can choose between either analysing the sentence differently, or allowing NPs, VPs, etc. to be discontinuous. There are two very good reasons for opting for the latter possibility.

Firstly, the syntactician in the generative tradition has the task to describe in her grammar ultimately all patterns of words that form correct sentences; the syntactician tries to achieve this not just in any odd way, but in a way which reveals syntactic generalizations. From this point of view it is attractive to be able to express a generalization like 'An S consists of an NP and a VP' without implying that the NP and VP must be continu-

[2] Or rather, *multisets* of words, i.e. unordered collections, possibly with duplicates.

ous. Allowing phrases to be discontinuous thus makes it possible to express stronger generalizations about constituency at higher levels.

Secondly, components like NP, PP and AP have a clear semantic status, and are therefore attractive building blocks in a compositional semantics. The semantic argument is partly one of good taste, since, according to the compositional view, the words making up a phrase have an independent meaning of their own that can in principle be used instead of the phrase meaning. The semantic argument reinforces the syntactic generalization argument, though: when discontinuous constituents are allowed, a semantic rule gets coupled to a syntactic rule that describes a larger class of structures, which is the more satisfactory since discontinuous phrases, like those caused by parentheticals, are often semantically equivalent to their continuous form.

Independent motivation for discontinuous structures in linguistic representations have also been given by Bach (1979, 1980) in his analysis of transitive verb phrases. Bach (1979) offers an analysis of control verbs in Montague Grammar in which various kinds of control verbs combine with complements to form transitive verb phrases. In the transformational account, verbs like *persuade* and *consider* share the common designation as 'control verbs' since their NP objects control the subject of their sentential complements. Bach classifies control verbs lexically (cf. (2)) as verbs which combine with controlled complements to form transitive verb phrases (TVPs) that lack only the controlling complement to form intransitive verb phrases (IVPs).

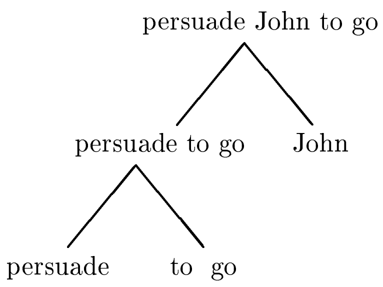
(2)	persuade to go	–	Vt/ViP + ViP
	consider competent	–	Vt/AdjP + AdjP

Since the controlled complement is combined first with the control verb, the combination of the resulting TVP with its NP complement cannot happen as the product of simple concatenation. Rather, a discontinuous operator called 'Right-Wrap' (RWRAP) is used to form an IVP by sticking the NP into the TVP right after the head. This operator is formalized in (3).

(3)	RWRAP(a,b)	=	a + b	if a is simple
			X + b + W	if a has the form $[_{XP}$ X W]

RWRAP (with + defined as concatenation) gives structures like the one depicted in (4), which have a clear interpretation in terms of trees with crossing branches.[3]

(4)



Bach takes advantage of the argument structure allowed by representing the combination of discontinuous TVPs with their objects to characterize passivization as a rule which applies solely to TVPs. This accounts for the constraints on passivization of control verbs relative to raising verbs in accord with the fact that, among lexical classes of transitive and intransitive verbs (belonging to the phrasal classes by default), only the transitive verbs passivize. The overall simplicity of this account provides a srong argument in favour of representing discontinuous constituents.

Jacobson (1987) accepts Bach's analysis with RWRAP and uses it to also characterize verb-particle constructions as TVPs, as they occur in sentences like (5).

(5) *Mary will wake me up at seven-thirty.*

In cases like this, *wake* combines with its particle to form a TVP and the NP object is wrapped into the TVP to form an IVP. Jacobson recasts the RWRAP operation in the GPSG framework as context-free promotion of the SLASH feature.

(6) $VP \quad \Rightarrow \quad V_1\ NP\ TVP[SL : V_1]$

Promotion of the SLASH feature is used to account for the discontinuity

[3] See Vogel & Bunt (1992) for a discussion of the relation between Right-Wrap and a restricted form of the Discontinuous Phrase Structure Grammar formalism, described below, called $DPSG^R$.

created by relative clause extraposition as well as that represented by crossing branches in tree representations.

The work of all three, McCawley, Bach, and Jacobson, suggests that there is sufficient linguistic motivation to maintain the constituent structure implied by discontinuous tree representations, in spite of the tendency to avoid drawing trees with crossing branches.

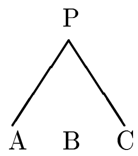
This paper does not so much present a *theory* about discontinuous constituency, but it develops certain *tools* which are simple yet quite powerful for the description of discontinuous structures, and which can be used in a wide variet of grammar formalisms and their computer implementations. Central is the notion of a *discontinuous tree*, which is a tree-like structure describing discontinuous constituency. A second important notion is that of *adjacency sequence*, which generalizes the notion of adjacency pair (or left-right neighbourship) in a way that applies to structures with discontinuities. This latter notion makes it possible to formulate rewrite rules which describe discontinuous structures. We outline the incorporation of such rules in a (generalized) phrase-structure grammar; the resulting formalism we call *Discontinuous Phrase-Structure Grammar*, or *DPSG*. We briefly consider an ID/LP format for DPSG which allows us to express strong generalizations in terms of ID rules.

We also briefly consider the possible use of discontinuous trees in Categorial Grammar and Head-driven Phrase Structure Grammar. We conclude with a few words on algorithmic complexity and on the computer implementation of DPSG for purposes of parsing and generation.

2 Trees with discontinuities

When we want to represent that a phrase P has constituents A and C, while there is an intervening element B, we must allow the node corresponding to P to dominate the A and C nodes without dominating the B node, even though this node is located between A and C (7). This will have the consequence that the structure gets crossing branches, if we still want every node to be dominated by a higher node. In what respects exactly do these structures differ from ordinary trees? McCawley (1982) has tried to answer this question, suggesting a formal definition for trees with discontinuities by amending the definition of an ordinary tree.

(7)



An ordinary tree is often defined as a set of elements, called NODES, on which two relations are defined, immediate dominance (D) and linear precedence ($<$). These relations are required to have certain properties, to the effect that a tree has exactly one root (or 'top') node, which dominates every other node; that every node in a tree has exactly one 'mother' node, etc. (see e.g. Wall, 1972). Two of the required properties that are worth considering here, are the following.

First, the precedence relation has the property that a nonterminal node x precedes a node y if and only if every node dominated by x precedes every node dominated by y:

(8) For any two nodes x and y in a tree, x $<$ y iff for all nodes u and v, if x dominates u and y dominates v, then u $<$ v.

Second, any two nodes are required to either dominate or precede one another, but not both:

(9) For any two nodes x and y in a tree, then either x D' y, or y D' x, or x $<$ y, or y $<$ x. (D' designates the relation of dominance: the reflexive and transitive closure of D.)

Together, (8) and (9) have the effect of excluding discontinuities in a tree. For suppose a node x would dominate nodes y and z without having a dominance relation with node w, where y $<$ w $<$ z. Since neither x dominates w nor w dominates x, by (9) either x $<$ w or w $<$ x. But x dominates a node to the right of w, so by (8) x does not precede w; and w is to the right of a node dominated by x, so w does not precede x either. A structure like (7) is thus excluded, since P neither dominates nor precedes B.

McCawley's definition of trees with discontinuities consists of a list of seven axioms like (8) and (9) which include, instead of (9), the requirement that a node has no precedence relation to any node it dominates:

(10) For any two nodes x and y in a tree, if x *D*' y then neither x $<$ y nor y $<$ x.

McCawley's definition is inaccurate in some respects; it is redundant, and it unintentionally allows structures to be cyclic or not wholly connected. An attractive alternative to an 'axiomatic' definition of tree-like structures is a recursive one, which is especially attractive in connection with top-down generation. Ordinary trees are very easy to define recursively since their building blocks are again trees, if we consider terminal nodes as atomic trees. Such a definition may run as follows:

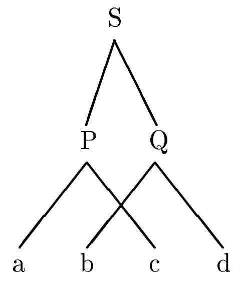
(11) Let N be a non-empty set whose elements are called NODES.

- (i) If t $\in$ N, then <t,[ ]> is a tree, where [ ] represents the empty list;
- (ii) If X_1, X_2, .., X_k are disjoint trees and a $\in$ N, then <a, [X_1,X_2, ..,X_k]> is a tree.
  Instead of '<a,[X_1,X_2,..,X_k]>', we shall also write more simply: 'a(X_1,X_2,..,X_k)'.
- (iii) No other structures than those defined by (i) and (ii) are trees.

According to this definition, a tree is an ordered pair consisting of a node and a list. The node is said to be the top of the tree and to dominate the members of the list. The top node is also said to be the 'mother' of the members of that list, which are called the 'daughters'. A node in a tree <t,[ ]> dominates zero subtrees, and is called a *terminal node*.

A recursive definition of a tree with discontinuities, or 'DISCOTREE', is not so easily given, since such a structure is built up of substructures which themselves are not necessarily (disco-)trees. Example (12) illustrates this.

(12)

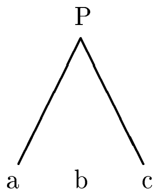


'Crossing discontinuities', which would give rise to structures like (12), do occur quite easily; examples are provided in (13). (In these examples, one of the crossing discontinuous constituents is underlined; the other italicized. Example a is from Williams (1974); b is from Stucky (1987); c is from Dowty (1992).)

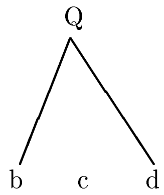
(13) a. *Everybody* is so strange *whom I like* that I can't go out in public with them.
b. An impeccably dressed man struck up *a conversation* with me on the plane last month *about 'Situations and Attitudes'* who was going to Missoula, Montana.
c. *Some guy* was hired *that Sue knew* to fix the plumbing.
d. *He invited* the vice-chairman, *I think it was*, of the committee.

The discotree S in (12) is made up of the substructures with top nodes P and Q:

(14a)



(14b)



These structures are not well-formed discotrees since they contain 'loose' nodes (b and c), not connected to the rest of the structure. Therefore, we first define these substructures ('SUBDISCOTREES') separately; this can be done recursively.

(15) Let N be a non-empty set whose elements are called NODES.

(i) If $t \in N$, then the pair $<t,[\ ]>$ is a subdiscotree; such a subdiscotree is called ATOMIC;

(ii) If $x \in N$ and $X_1, .., X_k$ are subdiscotrees that do not share any subdiscotrees[4] then the pair $<x, [X_1, Y_2, .., Y_j, X_k]>$ is a subdiscotree, where $Y_i = X_i$ or Y_i a sub-list of $\{X_2, .., X_{k-1}\}$.

[4]Two subdiscotrees X and Y share a subdiscotree T if both X and Y contain a node dominating T.

(iii) No other structures than those defined by (i) and (ii) are subdiscotrees.

Instead of '$<x, [X_1, .., [..], .., X_k]>$', we also write, more simply: '$x(X_1, X_2, .., [..], .., X_k)$'. The idea is that a subdiscotree consists of a top node x and a sequence of nodes dominated by x (daughter nodes), possibly interrupted by lists $[X_i, X_j, ..]$ of nodes *not* dominated by x; these are called INTERNAL CONTEXT of x, or CONTEXT DAUGHTERS of x.

For example, the structure in (14a) can be viewed as a graphical representation of the subdiscotree written in full and in simplified form in (16):

(16) $X = <P, [a, [b], c]> = P(a, [b], c)$

Similarly, the graph (12) is a visualization of (17):

(17) $X = <S, [<P, [a, [b], c]>, <Q, [b, [c], d]>]>$
$= S(\ P(a, [b], c), Q(b, [c], d)\ )$

We define the notions of dominance and direct dominance for subdiscotrees as follows:

(18) Direct dominance (D):

(i) in an atomic subdiscotree $<x, [\ ]>$, there is no node that is dominated by node x;

(ii) in a subdiscotree $<x, [X_1, Y_1, .., Y_j, X_k]>$ the node x directly dominates the top nodes of X_1, of X_k, and of every Y_i that is a subdiscotree (but not of those Y_i that are a list of subdiscotrees; a node does not dominate its context daughters).

Dominance (D^*) is defined as usual, as the reflexive and transitive closure of direct dominance.

A discontinuous tree is now simply a subdiscotree without any 'loose' nodes (nodes that are not connected to the rest of the structure through the dominance relation). In other words, every node in a discotree has a mother node.

(19) Discontinuous tree (DISCOTREE):
A discontinuous tree is a subdiscotree X where every node is dominated by some other node in X.

Since the top node of a (sub)discotree dominates all the nodes which have a mother node, it follows that the top node of a discotree dominates every node in the structure.

We now have a reliable definition of discontinuous tree structures. The next point to consider is how these structures can be generated by a grammar. We know that ordinary tree structures are generated by phrase-structure grammars; can discontinuous trees also can be produced by some sort of phrase-structure rules? This question (which to our knowledge has not been addressed before) turns out to be far from trivial; to answer it, we will have to look closely at linear precedence and adjacency in discotrees.

2.1 Linear precedence and adjacency in discontinuous trees

A phrase-structure rule rewrites a constituent into a sequence of pairwise adjacent constituents. The formulation of phrase-structure rules for discontinuous trees thus requires a notion of adjacency in such structures. Since adjacency is a special case of linear precedence, we first have to define the precedence relation for discotrees. We may try to do this in the same way as usual for ordinary trees.

In the axiomatic definition of trees, the set of terminal nodes is assumed to be ordered and the precedence relation is defined in a 'bottom-up' way on the basis of this ordering (see (8)).

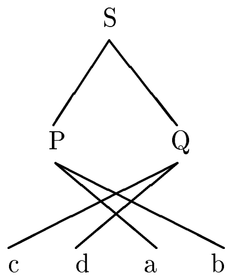
In the recursive approach, where trees are defined in a 'top-down' fashion, no ordering of the terminal nodes is assumed (see (15)). Instead, linear precedence ($<$) is defined recursively in a top-down way by (20):

(20) (i) In a tree $\mathrm{A}(\mathrm{X}_1,.., \mathrm{X}_n)$, $\mathrm{X}_i < \mathrm{X}_j$ iff $\mathrm{i} < \mathrm{j}$.

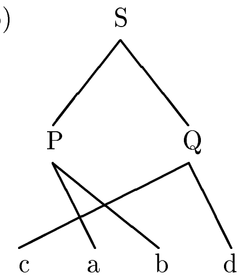
(ii) If P, Q, x and y are nodes in a tree such that $\mathrm{P} < \mathrm{Q}$, P D x, and Q D y, then $\mathrm{x} < \mathrm{y}$.

The first clause in this definition defines a partial ordering between 'sister' nodes. The second clause makes this ordering total, and rules out weird structures like (21).

(21a)



(21b)



It is easily verified that the precedence relation defined in this way is transitive, as an ordering should be. The obvious disco-counterpart of this definition is:

(22) (i) In a subdiscotree $A(X_1, .., [.., X_i, ..] ,.., X_n)$, $X_i < X_j$ iff $i < j$.

(ii) If P, Q, x and y are nodes in a subdiscotree such that $P < Q$, P D x, and Q D y, then $x < y$.

Adjacency is now simply precedence without intervening nodes:

(23) two nodes x and y in a discotree are adjacent iff $x < y$ and there is no z such that $x < z < y$.

We shall write '$x + y$' to indicate that x and y are adjacent.

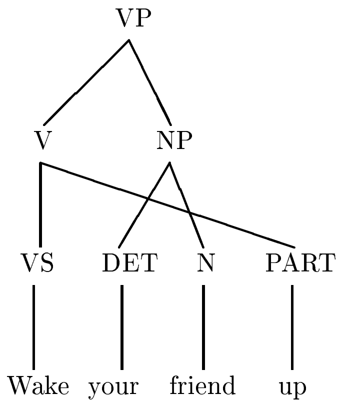
Unfortunately, this notion of adjacency is inadequate for formulating grammar rules that operate with discontinuous constituents. We turn to the issue of grammar rules for generating discontinuous trees in the next section.

3 Discotrees in grammar rules

3.1 Adjacency and concatenation

The following example shows that the notion of adjacency, introduced above, is not adequate for use in rewrite rules. Suppose we want to generate the discontinuous tree structure (24):

(24)



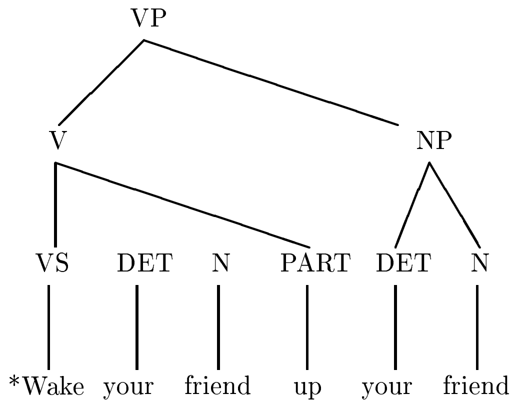
To generate this structure, we need rules like the following, where square brackets indicate internal context elements:

(25)	VP	→	V + NP
	V	→	VS + [DET] + [N] + PART
	NP	→	DET + N
	VS	→	wake
	DET	→	your
	N	→	friend
	PART	→	up

According to the second clause in (22), however, the particle *up* precedes the determiner and the noun, so these rules generate the incorrect structure (26). This problem occurs generally when a discontinuous constituent is to be concatenated with some other constituent. We thus need a different precedence relation for discontinuous trees.

The source of the problem is that, by the second clause in (22), a node x which is an internal context daughter of node P and a 'real' daughter of another node Q, either precedes or is preceded by all the daughter nodes of P, depending on whether P precedes Q or Q precedes P. But characteristic of an internal context node is precisely that it is located *in between* the 'real' daughters of a node (of node P, in this case). We therefore relax the second clause in (22) to (27).

(26)



(27) if X and Y are nodes in a subdiscotree such that $X < Y$, then X's leftmost daughter precedes Y's leftmost daughter.

The notion 'leftmost daughter' *Lm* (X) of X is defined by (28):

(28) (i) if X is an atomic subdiscotree $<\mathrm{x}, [\;]>$, then $Lm(\mathrm{X}) = \mathrm{x}$;

(ii) else X is a subdiscotree of the form $<\mathrm{x}, [\mathrm{X}_1,.., [..]\ ,.., \mathrm{X}_n]>$ and $Lm(\mathrm{X}) = \mathrm{X}_1$.

With this notion of precedence, and the notion of adjacency that follows from it, the rules (25) give besides the incorrect structure (26) also the correct possibility (24). To prevent the generation of incorrect structures like (26), we define adjacency slightly differently:

(29) Two nodes x and y are adjacent iff:

(i) $Lm(\mathrm{x}) < Lm(\mathrm{y})$;

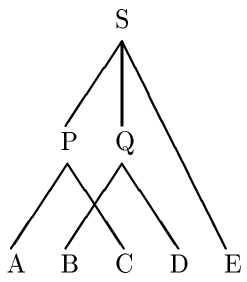
(ii) for every node z such that $Lm(\mathrm{x}) < \mathrm{z} < Lm(\mathrm{y})$: x D z.

In other words, x and y are adjacent if x precedes y and y's leftmost daughter is the first node to the right of x's leftmost daughter which does not belong to x. With this definition of adjacency the rules (25) do not generate the structure (26), since the V and NP in that case are not adjacent.

3.2 Discotrees in Phrase-Structure Grammar

Upon closer inspection, the adjacency relation as now defined is still not quite satisfactory from the point of view of formulating rewrite rules. The following example illustrates this.

(30)



Suppose we want to generate the structure (30). To generate the S node, we would like to write a phrase-structure rule that rewrites S into its constituents, like (31):

(31) S $\rightarrow$ P + Q + E

But this rule would not apply here, since Q and E are not adjacent; according to (29), Q has C as its right neighbour, not E. Therefore, the correct rule for generating (30) would be (32):

(32) S $\rightarrow$ P + Q + [C] + [D] + E

This is ugly, and even uglier rules are required in more complex cases with discontinuities at different levels. Moreover, there seems to be something fundamentally wrong, since the C and D nodes are on the one hand internal context for the S node, according to rule (32), while on the other hand they are also dominated by S. That is, these nodes are both 'real' constituents of S and internal context of S.

To remedy this we introduce a new concept called ADJACENCY SEQUENCE, which generalizes the traditional notion of sequence of adjacency pairs. The definition goes as follows:

(33) A sequence <a, b,..., n> is an adjacency sequence iff:

(i) every pair <i,j> in the sequence is either an adjacency pair or is connected by a sequence of adjacency pairs of which all members are a constituent of some element in the subsequence <a, b,..., i>;

(ii) the elements in the sequence do not share any constituents.

For example, in the structure (30) the triple <P, Q, E> is an adjacency sequence since <P, Q> is an adjacency pair and Q and E are connected by the sequence of adjacency pairs Q-C-D-E, with C and D constituents of P and Q, respectively. Moreover, P, Q, and E do not share any substructures. The triple <P, B, C>, on the other hand, is not an adjacency sequence since P and C share the constituent C.

This notion of adjacency sequence can now be used to define phrase-structure rules for discontinuous trees as prescriptions to rewrite a nonterminal into a sequence of constituents which forms an adjacency sequence, like (31), where some of the elements (but not the first and not the last) may be marked as internal context elements. A phrase-structure grammar consisting of rules of this kind we call a *Discontinuous Phrase-Structure Grammar (DPSG).*

A DPSG rule A $\rightarrow$ B + C generates exactly those structures where the immediate daughters of B and C are interwoven, or 'sequence-unioned' (Reape, 1990), with the additional constraint that B's leftmost daughter precedes C's leftmost daughter.

It is worth emphasizing that this notion of phrase-structure rule is a generalization of the standard notion, since an adjacency sequence as defined by (33) subsumes the traditional notion of sequence of adjacency pairs. We have also seen that trees with discontinuities are a generalization of the traditional tree concept. Therefore, phrase-structure rules of the familiar sort coincide with DPSG rules without discontinuous constituents, and they produce the familiar sort of trees without discontinuities. In other words, DPSG rules can simply be added to an ordinary PSG (including a generalized or augmented PSG), with the result that the grammar generates trees with discontinuities for sentences with discontinuous constituents, while doing everything else as before.

3.3 Discotrees in Categorial Grammar

Categorial Grammars do not aim at the explicit generation of constituent structures, but they are based on string concatenation; therefore, it can hardly come as a surprise that discotrees can be introduced in CGs just as easily as in PSGs.

The following example illustrates this possibility. Consider first the simple sentence

(34) *John kissed Mary.*

To analyse this in a CG-fashion, the following categories may be assigned to the individual lexical items:

```
(35)    John  : NP
        kissed: (NP\S)/NP
        Mary  : NP
```

Right concatenation of the verb with the object-NP gives a verb phrase of category `NP\S`, which concatenates to the left with the subject-NP to form an S.

Now consider the similar sentence with a discontinuity in the verb phrase:

(36) *John woke Mary up.*

To generate this sentence, we keep the NPs as before but assign the following categories to the verb and the particle:

```
(37)    woke: (NP\S)[/NP]/PART
        up  : PART
```

The notation [..] has been used here, as in the case of discontinuous PSG rules, to indicate that the material inside the brackets is internal context. To make this work, all we have to do is amend the right concatenation rule of CG to one that allows internal context:

```
(38)    A[/C]/B + [C] + B ==> A/C
```

This says that an element of category A[/C]/B combines with an element of category B occurring to its right while there is an intervening element of category C. The resulting element combines with a C on its right to form an A. (This could, of course, be generalized for sequences of more than one intervening elements.) Application of the scheme (38) to (36) and (37) gives the reduction of *woke .. up* to a discotree, with category `(NP\S)/NP`, and the rest goes the same as before.

For a really satisfactory treatment of cases like sentence (36) we should, of course, take into account that the particle which 'concatenates' with the verb is the one that the verb is expecting, and not just any particle. In this respect, the situation is the same as in PSG rules, and the solution is again to add features and unification to the grammar; see for instance Van der Linden (1989).

3.4 An ID/LP format for discontinuous rules

Sofar, we have considered the use of discotrees in rewrite rules only for their classical form, in which they state something about immediate dominance as well as about linear precedence. One of the important innovations of GPSG has been the decoupling of these aspects, which makes it possible to formulate rules that express much greater generalizations.

A rule that generates a discontinuous tree must by its very nature say something about immediate dominance as well as about linear precedence, so it seems, and an ID/LP format for such rules therefore might appear strange.

This is a wrong impression, however. A rule like (39) is equivalent to the ID/LP rules (40), where CD stands for (immediate) context daughters:

(39) $\mathrm{R} \rightarrow \mathrm{A} + [\mathrm{B}] + [\mathrm{C}] + \mathrm{D}$

(40) ID: $\mathrm{R} \rightarrow \{\mathrm{A}, \mathrm{D}\}$
CD: $\{\mathrm{B}, \mathrm{C}\}$
LP: $\mathrm{A} < \mathrm{D}$; $\mathrm{B} < \mathrm{C}$.

In this example, the ID/LP-like formulation clearly has no advantages whatsoever over the traditional formulation; such advantages can only be expected if the same dominance rule allows various linear orders. This is precisely what is often the case with discontinuities; for example, in Dutch an intervening adverb (phrase) can be placed on a variety of positions. If, in the case of example (39), the B and C constituents may appear in arbitrary order, the ID/LP-like format becomes more attractive:

(41) ID: $\mathrm{R} \rightarrow \{\mathrm{A}, \mathrm{D}\}$
CD: $\{\mathrm{B}, \mathrm{C}\}$
LP: $\mathrm{A} < \mathrm{D}$.

The LP rul has now become very simple, since internal context elements by definition can only stand in between the 'real' daughters of the top node.

The exact ID/LP-like format of DPSG rules is rather complex, when more 'real' or context daughters are involved, features are taken into account and semantics is added; the reader is referred to Bunt (1990) for more details.

The strict separation of dominance and precedence is also one of the features of Head-driven Phrase Structure Grammar (HPSG; see Pollard & Sag, 1987; 1991). For bounded discontinuities, which is what the DPSG concepts are useful for, HPSG like GPSG faces the problem of having to specify the bounded internal contexts which may separate the parts of a discontinuous constituent. In HPSG we can do this by 'simulating' the use of discotrees as follows.

HPSG does not build phrase structure representations of the usual kind, but builds attribute-value matrices where some of the attributes represent dominance relations. More precisely, the attributes HEAD-DAUGHTER, COMPLEMENT-DAUGHTERS, ADJUNCT-DAUGHTER, FILLER-DAUGHTER are used to represent mother-daughter relations with certain functional characteristics. We can add to this an attribute corresponding to the 'context daughter' relation of DPSG: CONTEXT-DAUGHTER, and thus build feature structures corresponding to (sub)discotrees, useful in parsing or generation of sentences with bounded discontinuities. Expectations (or requirements) concerning the presence of certain types of daughters in certain types of phrases are expressed in HPSG by means of attributes whose values propagate according to certain general principles, like the Subcat Principle. Expectations concerning complement daughters are expressed in the values of the SUBCAT attribute, those concerning filler daughters in the SLASH attribute, and those concerning head-adjunct daughter combinations either in the ADJUNCTS feature (Pollard & Sag, 1987) of the head duaghter or in the MOD feature of the adjunct daughter (Pollard & Sag, 1991). Similarly, for internal context daughters we introduce a feature whose values propagate in a way more or less similar to the SUBCAT feature. The details of how this might work are beyond the scope of the present paper.

4 Complexity and computer implementation of DPSG

DPSG obviously has greater than context-free generative capacity; on the hand, it has restrictions that make it less powerful than context-sensitive grammar. Exactly where DPSG sits in between these two has not yet been established precisely. Vogel & Bunt (1992) and Vogel & Erjavec (1994) argue that, especially when DPSG is restricted slightly in the kinds of crossing branching it allows,[5] it has properly less power than the Mildly Context-Sensitive Grammars (Joshi, Vijay-Shanker & Weir, 1989).

A first computer implementation of DPSG has been made within the context of the TENDUM dialogue project (see Bunt *et al.*, 1985). A parser-interpreter has been built, based on the chart parsing technique,[6] where the syntactic side of DPSG that has been outlined here, is combined with a somewhat Montague-like semantic theory, called *two-level model-theoretic semantics* (Bunt, 1985), and with a pragmatic theory which construes the communicative functions of utterances in dialogues in terms of the kind of information transfer they accomplish upon correct understanding (Bunt, 1989).[7] The output of the parser consists of one or more triples, depending on ambiguity in the input, each formed by

1. a syntactic analysis in the form of a discotree;
2. a semantic representation in the form of an expression of the logical language EL/F, a type-logical language based on Ensemble Theory, an extension of standard set theory; see Bunt (1985);
3. a pragmatic analysis in the form of a bundle of features, relevant for the determination of the input's communicative functions.

The parsing algorithm is based on the active chart parsing concept; it uses a special matrix-driven strategy for the application of rules, and it incorporates additional bookkeeping devices for the correct handling of context daughters and the n-ary adjacency-sequence relation of DPSG. Above, we have already emphasized that trees with discontinuities are a generalization of the traditional tree concept, and that the DPSG notion of phrase-structure rule is a generalization of the standard notion. As a result, phrase-structure rules of the traditional sort coincide with DPSG rules without discontinuous constituents, and they produce the familiar sort of trees without

[5] The restriction considered in these studies is that any node c, which occurs as a context daughter of a node t, must also be directly dominated by some node s, distinct from t, that bears a 'generalized dominance' relation to t. The 'generalized dominance' relation combines the ordinary dominance relation with that of the relation between a node and its context daughters.

[6] See Bunt, Thesingh & Vander Sloot (1987).

[7] For developments in this pragmatic theory since the TENDUM project see Bunt (1990; 1994).

discontinuities. A PSG parser equipped with the additional bookkeeping devices mentioned above can thus process a grammar consisting partly of ordinary PSG rules and partly of DPSG rules, producing syntactic (disco-)trees with discontinuities for sentences with discontinuous constituents, while operating in the usual way for sentences without discontinuities. The reader is referred to Bunt (1991) for a description of this chart parser.

The use of DPSG in the generation of sentences, starting from formal representations of semantic content and communicative function, has been explored both theoretically and in an experimental computer implementation discussed in Bunt (1987).

An interesting recent development is the expression of feature structures and DSPG rules in an extended version version of the semantic representation language ELF/F, mentioned above. This allows the expression of syntactic and semantic information in a single representation formalism, and gives a formal, model-theoretic interpretation of the syntactic part of a DPSG grammar. The extended representation language is called 'GEL', for Generalized Ensemble Language. In contrast with the traditional view of phrase-structure rules as recipes for building phrase structures, and with the declarative view in the parsing-as-deduction paradigm as a proposition which is true if the conditions for building the phrase structure are satisfied, a rule in GEL is best viewed as a proposition in Dynamic Semantics: it can be evaluated recursively and evaluates not to true or false, but to the minimal change in the model[8] needed to make the proposition true. This idea and its implementation in a chart parser for DPSG are described in Bunt & Van der Sloot (1993).

References

Bach, E. (1979) Control in Montague Grammar. *Linguistic Inquiry 10*, 515-531.

Bach, E. (1980) In Defense of Passive. *Linguistics and Philosophy 3*, 297-341.

Bunt, H.C. (1985) *Mass terms and model-theoretic semantics.* Cambridge University Press, Cambridge, England.

Bunt, H.C. (1987) Utterance generation from semantic representation augmented with pragmatic information. In G. Kempen (ed.) *Natural language generation.* Kluwer/Nijhoff, The Hague.

[8]That is, in this case, in the chart.

Bunt, H.C. (1989) Information dialogues as communicative action in relation to partner modelling and information processing. In M.M. Taylor, F. Neel, & D.G. Bouwhuis (eds.) *The Structure of Multimodal Dialogue.* North-Holland Elsevier, Amsterdam, 48-73.

Bunt, H.C. (1990) DIT - Dynamic Interpretation in Text and Dialogue. In L. Kalman & L. Polos (eds.) *Papers from the Second Symposium on Logic and Language.* Akademiai Kiado, Budapest.

Bunt, H.C. (1991) DSPG and its use in parsing. In M. Tomita (ed.) *Parsing Technologies.* CMU, Pittsburgh.

Bunt, H.C. (1994) Context and Dialogue Control. *THINK Quarterly 3 (1)* , 17-29.

Bunt, H.C., Beun, R.J., Dols, F.J.H., Linden, J.A. van der, & Schwartzenberg, G.O. thoe (1985) The TENDUM dialogue system and its theoretical basis. IPO Annual Progress Report 19, 105-113

Bunt, H.C., Thesingh, J.C. & Sloot, K. van der (1987) Discontinuities in trees, rules and parsing. *Proceedings of the Third Conference of the European Chapter of the ACL,* Copenhagen 1987, 203-211.

Bunt, H.C. & Sloot, K. van der (1993) Parsing as Dynamic Evaluation. In *Proceedings of the Third International Workshop on Prasing Technologies IPWT'93*, ITK, Tilburg 1993, 27-37.

Dowty, D.R. (1994) Toward a Minimalist Theory of Syntactic Structure. *This volume.*

Emonds, J.E. (1979) Appositive relatives have no properties. *Linguistic Inquiry*, vol. 10, 211-243.

Gazdar, G., Klein. E., Pullum, G.K. & Sag, I.A. (1985) *Generalized Phrase Structure Grammar.* Harvard University Press, Cambridge, Massachusetts.

Linden, E.-J. van der (1989) Lambek Theorem Proving and Feature Unification. In: *Proceedings of the 4th EACL Conference, Manchester, April 1989.*

Jacobson, P. (1987) Phrase Structure, grammatical relations, and discontinuous constituents. In G.J. Huck & A.E. Ojeda (eds.) *Discontinuous Constituency*, Academic Press, New York etc.

McCawley, J.D. (1982) Parentheticals and Discontinuous Constituent Structure. *Linguistic Inquiry*, vol. 13, no. 1, 91-106

Pollard, C.H. & Sag, I.A. (1987) *Information-based Syntax and Semantics*, Vol. 1. CLSI, Stanford.

Pollard, C.H. & Sag, I.A. (1991) *Topics in Constraint-Based Syntactic*

Theory. Universität des Saarlandes, Saarbrücken.
Reape, M. (1994) Getting things in order. *This volume.*
Stucky, S. (1987) Configurational Variation in English: A Study of Extraposition and Relation Matters. In: G. Huck & A. Ojeda (eds.) *Discontinuous Constituency. Syntax and Semantics 20.* Academic Press, New York etc.
Vogel, C.M. & Bunt, H.C. (1992) Representing Discontinuous Constituency: Motivations and Complexity. In *Proceedings of Pacific Rim International Conference on Artificial Intelligence*, Seoul, September 16-18, 1992.
Vogel, C.M. & Erjavec, T. (1994) Restricted Discontinuous Phrase Structure Grammar and Its Ramifications. In: C. Martin-Vide (ed.) *Current Issues in Mathematical Linguistics*, Elsevier, Amsterdam.
Wall, R.E. (1972) *Introduction to Mathematical Linguistics.* Prentice-Hall, Englewood Cliffs.
Williams, E. (1974) *Rule Ordering in Syntax.* MIT Linguistics Dissertation. Cambridge, Massachusetts.