DISCONTINUOUS CONSTITUENTS IN TREES, RULES, AND PARSING

Harry Bunt, Jan Thesingh and Ko van der Sloot
Computational Linguistics Unit
Tilburg University, SLE
Postbus 90153
5000 LE TILBURG, The Netherlands

ABSTRACT

This paper discusses the consequences of allowing discontinuous constituents in syntactic representions and phrase-structure rules, and the resulting complications for a standard parser of phrase-structure grammar.

It is argued, first, that discontinuous constituents seem inevitable in a phrase-structure grammar which is acceptable from a semantic point of view. It is shown that tree-like constituent structures with discontinuities can be given a precise definition which makes them just as acceptable for syntactic representation as ordinary trees. However, the formulation of phrase-structure rules that generate such structures entails quite intricate problems. The notions of linear precedence and adjacency are reexamined, and the concept of "n-place adjacency sequence" is introduced. Finally, the resulting form of phrase-structure grammar, called "Discontinuous Phrase-Structure Grammar", is shown to be parsable by an algorithm for context-free parsing with relatively minor adaptations. The paper describes the adaptations in the chart parser which was implemented as part of the TENDUM dialogue system.

1. Phrase-structure grammar and discontinuity

Context-free phrase-structure grammars (PSGs) have always been popular in computational linguistics and in the theory of programming languages because of their technical and conceptual simplicity and their well-established efficient parsability (Sheil, 1976; Tomita, 1985). In theoretical linguistics, it was generally believed until recently that natural language competence cannot be characterized adequately by a context-free grammar, especially in view of agreement phenomena and discontinuities (see e.g. Postal, 1964). However, in the early eighties Gazdar and others revived an idea, due to Harman (1963), of formulating phrase-structure rules not in terms of monadic category symbols, but in terms of feature bundles. With this richer conception of PSG it is not at all obvious whether natural languages can be described by context-free grammars (see e.g. Pullum, 1984). Generalized Phrase-Structure Grammar (GPSG; Gazdar et al., 1985), represents a recent attempt to provide a theoretically acceptable account of natural-language syntax in the form of a phrase-structure grammar.

Apart from being important in its own right, phrase-structure grammar also plays an important part in more complex grammar formalisms that have been developed in linguistics; in classical Transformational-Generative Grammar the base component was assumed to be a PSG; in Lexical-Functional Grammar a PSG is supposed to generate c-structures, and in Functional Unification Grammar context-free rules generate the input structures for the unification operation (Kay, 1979).

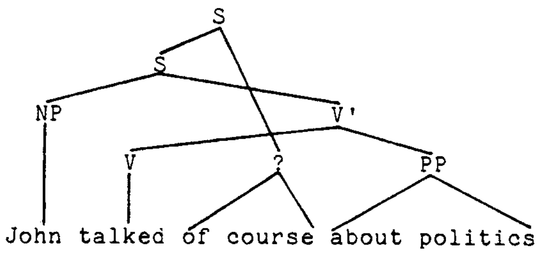
Phrase-structure grammar has one more attractive side, apart from its technical/conceptual simplicity and its computational efficiency, namely that it seems to fit the semantic requirement of compositionality very well. The compositionality principle is the thesis that the meaning of a natural-language expression is determined by the combination of (a) the meanings of its parts; (b) its syntactic structure. This entails, for a grammar which associates meanings with the expressions of the language, the requirement that the syntactic rules should characterize the internal structure of every expression in a "meaningful" way, which allows the computation of its meaning. In this way, semantic considerations can be used to prefer one syntactic analysis to another. PSGs area useful tool for the formulation of syntactic rules that meet this requirement, as phrase-structure rules by their very nature provide a recursive description of the constituent structure of complex expressions down to their smallest meaningful parts. However, PSG has one property that limits its applicability in describing constituent structure in natural language, namely that phrase-structure rules assume the constituents of an expression to correspond to adjacent substrings. In natural language it happens quite often, however, that the constituents of an expression are not adjacent. The English and Dutch example sentences (1)-(8) illustrate this. In (2)-(7) we see examples of major phrases, made up of parts that are not adjacent; so-called discontinuous constituents. We have discontinuous noun phrases in (5) and (7), a discontinuous adjective phrase in (3), discontinuous verb phrases in (1) and (4), and a discontinuous adverb phrase in (6).

(1) John talked, of course, about politics
(2) Which children did Anne expect to get a present from?
(3) This was a better movie than I expected
(4) Wake me up at seven thirty
(5) Will one of your cousins come who moved to Denmark?

(6) Leo is harder gegaan dan ooit tevoren (= Leo has been going faster than ever before)
(7) Ik heb een auto gekocht met 5 deuren (= I have bought a car with 5 doors)
(8) Ik hoor dat Jan Marie de kinderen de hond heeft helpen leren uitlaten (= I hear that John has helped Mary to teach the kids to walk the dog)

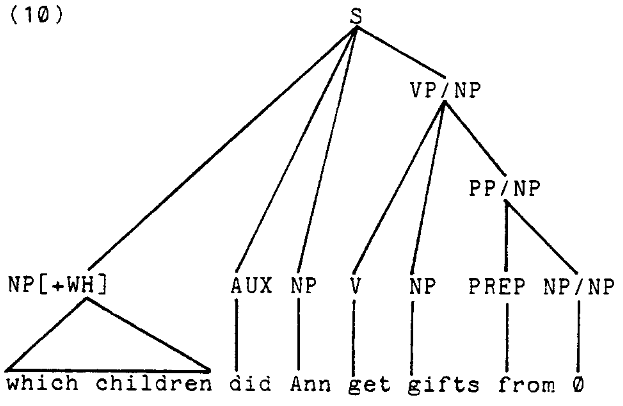
These examples do not represent a single class of linguistic phenomena, and it is doubtful whether they should all be handled by means of the same techniques.

Sentence (1), which has been discussed extensively in the literature, presents a problem for any analysis in terms of adjacent constituents, since the parenthetical "of course" divides the verb phrase "talked about politics" into non-adjacent parts. This means that we are forced to either consider the parenthetical as part of the VP, as Ross (1973) has suggested, or as a constituent at sentence level, as has been suggested by Emonds (1976; 1979). In the latter case, the sentence is analysed as consisting of the embedded sentence "John talked", with "of course" and "about politics" as specifiers at sentence level. McCawley (1982) provides detailed arguments showing that both suggestions are inadequate (which seems intuitively obvious, from a semantic point of view), and suggests, instead, the syntactic representation (9).

(9)


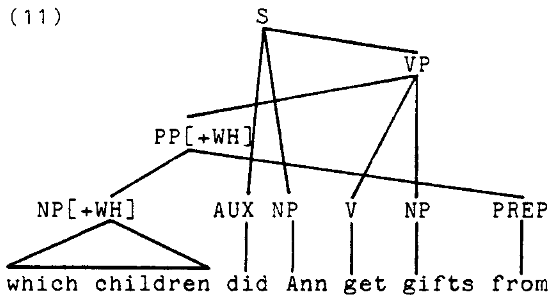
This is of course no longer an ordinary tree structure, but should that be a reason to reject it? McCawley takes the view that we should simply not be afraid of constituent structures like (9). We will return to this suggestion below.

Example (2) represents a different class of phenomena, which are conveniently thought of in terms of movements of parts of phrases. In this example, the NP "which children" can be thought of as having moved out of the PP "from which children", of which only the preposition has been left behind. In order to deal with such cases, in GPSG a special type of syntactic categories have been introduced, called "slash categories". For instance, the category PP/NP is assigned to a prepositional phrase which "misses" an NP. In the present example, this category would be assigned to "from". The assumption that an NP is missing propagates to higher nodes in the syntactic tree which the phrase-structure rules construct for the sentence, until it is acknowledged at the top level. Diagram (10) illustrates this.

(10)


If we want to do justice to the intuition that the sentence at surface level contains a constituent made up by "which children" and "from", we would have to draw a constituent diagram like (11), which, like (9), is no longer an ordinary tree structure.

(11)



The technique of using phrases that miss some constituent cannot be used for at least some of the examples (3)-(8), such as (5) and (7). In both these sentences the discontinuous NP contains a full-fledged NP, which cannot sensibly be said to "miss" the relative clause or prepositional phrase that occurs later in the sentence.

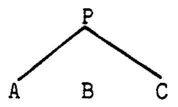
Whatever techniques may be invented to deal with such cases, it seems obvious that a grammar which recognizes and describes discontinuities in natural language sentences is a more suitable basis for semantic interpretation than one that squeezes constituent structures in a form in which they cannot be represented.

It therefore seems worth investigating the viability of tree-like structures with discontinuities, like (9) and (11).

2. Trees with discontinuities

If we want to represent the situation that a phrase P has constituents A and C, while there is an intervening phrase B, we must allow the node corresponding to P to dominate the A and C nodes without dominating the B, even though this node is located between the A and C nodes:

(12)



One consequence of allowing such discontinuities is that our structures get crossing branches, if we still want all nodes to be connected to the top node; (10) and (11) illustrate this. 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 a tree.

A tree is often defined as a set of elements, called "nodes", on which two relations are defined, **immediate dominance** (D) and **linear precedence** (<), which are required to have certain properties to the effect that a tree has exactly one root node, which dominates every other node (immediately or indirectly); that every node in a tree has exactly one "mother" node, etc. (see e.g. Wall, 1972).

Given the relations of immediate dominance and linear precedence, **dominance** is defined as the reflexive and transitive closure D' of D, and **adjacency** as linear precedence without intervening nodes.

A node in a tree is called **terminal** if it does not dominate any other node; the terminal nodes in a tree are totally ordered by the < relation. For nonterminal nodes the precedence relation satisfies the requirement that x < y if and only if every node dominated by x precedes every node dominated by y. Formally:

(13) for any two nodes x and y in the node set of a tree, x < y if and only if for all nodes u and v, if x dominates u and y dominates v, then u < v.

Part of the definition of a tree is also the stipulation that any two nodes either dominate or precede one another:

(14) for any two nodes x and y in the node set of a tree, either x D' y, or y D' x, or x < y, or y < x.

This stipulation has 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. By (14), either x < w or w < x. But x dominates a node to the right of w, so by (13) x does not precede w; and w is to the right of a node dominated by x, so w does not precede x either.

McCawley's definition of trees with discontinuities comes down to dropping the condition that any two nodes should either dominate one another or have a left-right relation. Instead, he proposes the weaker condition that a node has no precedence relation to any node that it dominates:

(15) for any two nodes x and y in the node set of a tree, if x D' y then neither x < y nor y < x.

We shall call a node u, situated between daughters of a node x without being dominated by x, **internal context** of x.

McCawley's definition of trees with discontinuities is inaccurate in several respects; however, his general idea is certainly correct: trees with discontinuities can be defined essentially by relaxing condition (14) in the definition of trees.

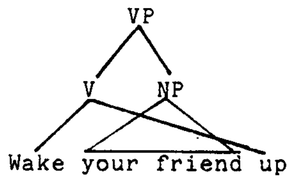
However, this is only the beginning of what needs to be done. The next question is how discontinuous trees can be produced by phrase-structure rules. This question, which is not addressed by McCawley, is far from trivial and turns out to have interesting consequences for the notion of adjacency in discontinuous tre
es.

3. Adjacency in phrase-structure rules for discontinuous constituents

A phrase-structure rule rewrites a constituent into a sequence of pairwise adjacent constituents. This means that we need a notion of **adjacency** in discontinuous trees, for which the obvious definition, given the < relation, would seem to be:

(16) two nodes x and y in the node set of a tree are adjacent if and only if 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 (or "neighbours"). A moment's reflection shows that this notion of adjacency unfortunately does not help us in formulating rules that could do anything with internal context constituents. The following example illustrates this. Suppose we want to generate the discontinuous tree structure:

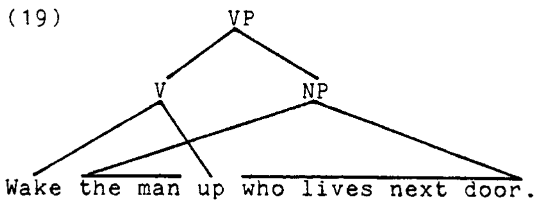
(17)


To generate the top node, we need a rule combining the V and the NP, like:

(18) VP --> V + NP

Since the V dominates nodes at either side of the NP, however, there is no left-right order between the NP and V nodes, leave alone a neighbour relation. For the same reason there would be no left-right relation between overlapping discontinuous constituents, as in (19).

These deficiencies can be remedied by replacing clause (14) in the definition of a tree by the more general clause (20).

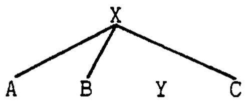
(19)


(20) A nonterminal node x in a tree is to the left of a node y in the tree if and only if x's leftmost daughter is left of y's leftmost daughter.

(We refrain here from a formal definition of "leftmost daughter" node, which is intuitively obvious.)

Note that (20) is indeed a generalization of the usual notion of precedence in trees, which could also be defined by (20). The recursion in (20) comes to an end since the terminal nodes are required to be totally ordered.

It should also be noted that (20) is not consistent with clause (14): by (20), we do get a precedence relation between a node and its daughter nodes (except the leftmost one) and internal context nodes. This is not quite unreasonable. In (21), for example, we do want that X < Y, and

(21)


since Y < C, that X < C, but not that X < B. We therefore adapt clause (14) to the effect that a mother node only precedes internal context nodes and daughter nodes which have internal context nodes to their left. Formally:

(22) For any nodes x and z in the node set N of a tree, if x D z and there are no nodes u,v in N such that x D u, not x D v, and u < v < z, then neither x < z nor z < x.

With the modifications (16) and (22), we have a consistent definition of "discontinuous trees" which allows us to write phrase-structure rules containing discontinuous constituents as follows:

(23) X --> A + B + [Y] + C

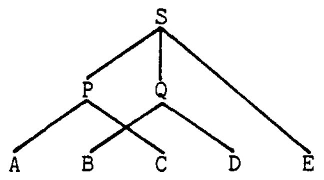
where the square brackets indicate that the NP is not dominated by the X node, but is only internal context. The "+" symbol represents the notion of adjacency, defined as before but now on the basis of te revised precedence relation "<":

(24) Two nodes x and y in a tree are adjacent if and only if x < y and there is no node z in the tree such that x < z < y.

Upon closer inspection, the neighbour relation defined in this way is unsatisfactory, however, as the following example illustrates.

Suppose we want to generate the following (part of a) tree structure:

(25)



To generate the S node, we would like to write a phrase-structure rule that rewrites S into its constituents, like (26):

(26) S --> P + Q + E

However, this rule would be of no help here, since P, Q and E do not form a sequence of adjacency pairs, as Q and E are not adjacent according to our definition. Rather, the correct rule for generating (25) would be (27):

(27) S --> P + Q + [C] + [D] + E

This is ugly, and even uglier rules are required in more complex trees 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 (27), 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 of **adjacency sequence**, which generalizes the traditional notion of a sequence of adjacency pairs. The definition goes as follows:

(28) A sequence (a, b,..., n) is an (n-place) adjacency sequence if and only if:
(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.[1]

For example, in the structure (25) 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. Another example of an adjacency sequence in (25) is the triple (P, B, D). The triple (P, B, C), on the other hand, is not an adjacency sequence, since P and C share the constituent C.

The use of this notion of adjacency sequence is now that the sequence of constituents, into which a nonterminal is rewritten by a phrase-structure rule, forms an adjacency sequence in this sense. The phrase-structure grammar consisting of rules of this kind we call **Discontinuous Phrase-Structure Grammar** or **DPSG**.[2]

It may be worth emphasizing that this notion of phrase-structure rule is a generalization of the usual notion, since an adjacency sequence as defined by (28) subsumes the usual 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 a classical PSG (including GPSG), with the result that the grammar generates trees with discontinuities for sentences with discontinuous constituents, while doing everything else as before.

4. DPSG and parsing

From a parser's point of view, a definition of adjacency as given in (24) is not sufficient, since it only applies to nodes within the context of a tree. A parser has the job of constructing such a set from a collection of substructures that may or may not fit together to form one or more trees for the entire sentence. Whether a number of subtrees fit together is not so easy if the end product may be a tree with discontinuities, since the adjacency relation defined by (20) and (24) allows neighbouring nodes to have common daughters. This is clearly undesirable. We therefore modify the definition (20) of adjacency by adding the requirement that two substructures (or their top nodes) can only have a precedence relation if they do not share any constituents:

(29) A node x in a collection of substructures for a potential tree (possibly with discontinuities) is to the left of a node y in the same collection if and only if x's leftmost daughter is left of y's leftmost daughter, and there is no node z which is shared by x and y.

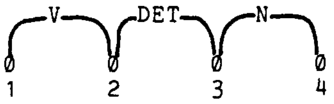
If the nodes x and y in this definition belong to the same tree, the additional requirement that x and y do not share any constituent is automatically satisfied, due to the "single mother" condition.

A parser for DPSG meets certain complications which do not arise in context-free parsing. To see these complications, we consider what would happen when a chart parser for context-free parsing (see Winograd, 1983) is applied to DPSG.

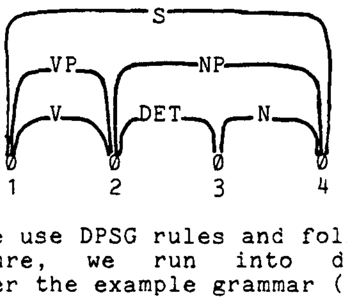
Context-free chart parsing is a matter of fitting adjoining pieces together in a chart. For example, consider the grammar:

(30) S --> VP NP
NP --> DET N
VP --> V

For the input "V DET N", a chart parser begins by initializing the chart as follows:

(31)


Given the arc V(1,2) in the chart, we look up all those rules which have a "free" V as the first constituent. These rules are placed in a separate list, the "active-rule list". We "bind" the V's in these rules to the V(1,2) arc, i.e. we establish links between them. When all constituents in a rule are bound, the rule is applied. In this case, the VP(1,2) will be built. This procedure is repeated for the new VP node. When nothing more can be done, we move on in the chart. The final result in this example is the chart (32).

(32)


When we use DPSG rules and follow the same procedure, we run into difficulties. Consider the example grammar (33).

(33) S --> VP + NP
NP --> DET + N
VP --> V + [NP] + PART

For the input "V DET N PART" the first constituent that can be built is NP(2,4); the second is VP(1,5). The VP will activate the S rule, but this rule will not be applied since the NP does not have a binding. And even if it did, the rule would not be applicable as the VP(1,5) and the NP(2,4) are not adjoining in the traditional sense.

In the next section we describe the provisions, added to a standard chart parser in order to deal with these difficulties.

5. A modified chart parser for DPSG

5.1 Finding all applicable rules

To make sure that the parser finds all applicable rules of a DPSG, the following addition was made to the parsing algorithm.

If a rule with internal context is applied, we first follow the standard procedure; subsequently we go through all those rules that appear on the active-rule list as the result of applying the standard procedure, giving bindings to those free constituents that correspond in category to the context-element(s) in the rule that was applied.

In the case of (33), this means that just before application of the VP rule (after the PART has been bound), we have the active-rule list (34). (Underlining indicates that a constituent is bound).

(34) VP --> $\underline{\text{V}}$ + [NP] + PART
VP --> $\underline{\text{V}}$ + [$\underline{\text{NP}}$] + PART
VP --> $\underline{\text{V}}$ + [$\underline{\text{NP}}$] + $\underline{\text{PART}}$

We now apply the rule building the VP. The standard procedure will add one rule to this list, namely S --> VP + NP. The VP is given a binding, so we obtain the following active-rule list:

(35) S --> $\underline{\text{VP}}$ + NP
VP --> $\underline{\text{V}}$ + [NP] + PART
VP --> $\underline{\text{V}}$ + [$\underline{\text{NP}}$] + PART
VP --> $\underline{\text{V}}$ + [$\underline{\text{NP}}$] + $\underline{\text{PART}}$

Since the VP-building rule contained an internal context element, the additional procedure mentioned above is now applied; a binding is given to the NP in (a copy of) the S rule. The S arc is now built in the chart, which does not cause any new rules to be added to the active-rule list. There are no free S's in the old active rule list either, which should be given a binding. So, we can look for other rules containing a free NP. There is one such rule, the second in (35), but this one will be neglected because it was already present in the rule list before; see (34). Note that it is essential that this rule is neglected, as there is already a version of the VP-rule on the active-rule list containing an NP with the same binding as the context-element.

It may also be noted that we have combined constituents in this example that are not adjoining in the traditional sense (i.e., in the sense of successive vertex numbers). In particular, we have applied the rule S --> VP(1,5) + NP(2,4). In a case like this, where the vertex numbers indicate that the constituents in a rule are overlapping, we must test whether these constituents form an adjacency sequence. This test is described below.

5.2 The adjacency sequence test

In order to make sure that only consituents are combined that form an adjacency sequence, the parser keeps track of daughter nodes and internal context in a so-called "construction list", which is added to each arc in the chart; internal context nodes are marked as such in these lists. Whether two (or more) nodes share a constituent, in the sense of common domination, is easily detected with the help of these lists.

By organizing these lists in a particular way, moreover, they can also be used to determine whether a sequence of constituents is an adjacency sequence in the sense of definition (28). This is achieved by ordering the elements in construction lists in such a way that an element is always either dominated by its predecessor in the list, or is internal context of it, or is a right neighbour of it. For instance, in the above example (25), P and Q have the construction lists (36):

(36) P:(A, [B], C)
     Q:(B, [C], D).

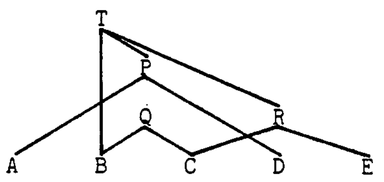
The rule S --> P + Q + E is now applicable, since the construction list for S would be the result of merging P's and Q's lists with that of E, which is simply E:(), with the result S:(A, B, C, D, E). From this list, it can be concluded that the triple (P, Q, E) is an adjacency sequence, since (P, Q) is an adjacency pair (since P's leftmost daughter, i.e. A, is adjacent to Q's leftmost daughter, i.e. B, as can be seen also in the construction lists), and Q and E are separated in S's construction list by the adjacency pair (C, D), whose elements are both daughters of P.

An example where the adjacency sequence test would give a negative result, is where the rule Y --> X + B + E is considered for a constituent X with construction list X:(A, [B], [C], D). The rule is not applicable, since the triple (X, B, E) would not form an adjacency sequence according to the construction list that the node Y would get, namely:

(37) Y:(A, B, [C], D, E).

The constituents B and E are separated in (37) by the sequence ([C], D), where C is marked as internal context; therefore, C is not dominated by either X or B, and hence the test correctly fails.

The currently implemented version of the DPSG parser is in fact based on a more restricted notion of adjacency sequence, where two constituents are viewed as sharing a constituent z not only if they both dominate z, but also if one of them dominates z and the other has an internal context node that dominates z (see note 1). This means that structures like (38) are not generated, since P and T would share node B, and T and R would share node C.

(38)


Note that a structure like (38) would be an ill-formed tree, since the nodes B and C violate the single-mother condition, and the nodes Q and R, moreover, are not connected to the root node.

To deal with this more restricted notion of adjacency sequence, the administration in the construction lists is actually slightly more complicated than described above.

6. **Conclusions**

Our findings concerning the use of discontinuous constituents in syntactic representations, phrase-structure rule, and parsers may be summarized as follows.

1. Tree-like structures with discontinuities can be given a precise definition, which makes them formally as acceptable for use in syntactic

representation as the familiar ordinary tree structures.

2. Discontinuous constituents can be allowed in phrase-structure rules generating trees with discontinuities, provided we give a suitable generalization to the notion of adjacency.
3. Trees with discontinuities are generalizations of ordinary tree structures, and phrase-structure rules with discontinuous constituents are generalizations of ordinary phrase-structure rules. Both concepts can be added to ordinary phrase-structure grammars, including GPSG, with the effect that such grammars generate trees with discontinuities for sentences with discontinuous constituents, while everything else remains the same.
4. Phrase-structure rules with discontinuities can be handled by a chart parser for context-free grammar by making two additions in the administration; one in the active-rule list for rules containing a discontinuous element to make sure that no parse is overlooked, and one in the arcs in the chart to check the generalized adjacency relation.

NOTES

1) In this paper, sharing a constituent has been taken simply as common domination of that constituent. An interesting issue is whether we should take sharing a constituent to include the following situation. A node x dominates a constituent z, while another node y is related to z in such a way that z is dominated by a node w which is internal context for y. (And still more complex definitions of constituent sharing are conceivable within the framework of DPSG.) Decisions on this point turn out to have far-reaching consequences for the generative capacity of DPSG. With the simple notion of sharing used in this paper, it is easily proved that DPSG is more powerful than context-free PSG, while further restrictions on the precedence relation in terms of constituent sharing may have the effect of making DPSG weakly equivalent to context-free grammar.

2) For applications of DPSG and a predecessor, which was called "augmented phrase-construction grammar", in syntactic/semantic analysis and automatic generation of sentences, the reader is referred to Bunt (1985; 1987).

ACKNOWLEDGEMENTS

I would like to thank Masaru Tomita for stimulating discussions about phrase-structure grammar and parsing in general, and DPSG in particular.

REFERENCES

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.

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.

Emonds, J.E. (1976) A transformational approach to English syntax. Academic Press, New York.

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

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

Harman, G. (1963) Generative grammars without transformaton rules: a defense of phrase structure. Language 39, 597-626.

Kay, M. (1979) Functional grammar. In Proc. Fifth Annual Meeting of the Berkeley Linguistics Society. Berkeley, CA, 142-158.

McCawley, J.D. (1982) Parentheticals and Discontinuous Constituent Structure. Linguistic Inquiry 13 (1), 91-106

Postal, P.M (1964) Constituent structure. Supplement to International Journal of American Linguistics 30.

Pullum, G.K. (1984) On two recent attempts to show that English is not a CFL. Computational Linguistics 10 (3/4), 182-187.

Ross, J.R. (1973) Slifting. In M. Gross, M. Halle & M.P. Schützenberger (eds.) The formal analysis of natural language. Mouton, The Hague.

Sheil, B. (1976) Observations on context-free parsing. Statistical Methods in Linguistics, 71-109.

Tomita, M. (1986) Efficient parsing for natural language. Kluwer Academic Publishers, Boston/Dordrecht.

Wall, R.E. (1972) Introduction to Mathematical Linguistics. Prentice-Hall, Englewood Cliffs.

Winograd, T. (1983) Language as a cognitive process. Addison-Wesley, Reading, MA.