Discontinuous constituency
Bunt, H.C.

Publication date: 1994

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright, please contact us providing details, and we will remove access to the work immediately and investigate your claim.
ITK Research Report No. 56
Discontinuous Constituency:
Introduction and Formal Tools
for Description and Processing
Harry Bunt

ITK
Warandelaan 2
P.O. Box 90153
5000 LE TILBURG
Harry.Bunt@kub.nl

November 1994

ISSN 0924-7807
Discontinuous Constituency: Introduction and Formal Tools for Description and Processing

Harry Bunt
Chapter 1

Discontinuous Constituency: Introduction

1.1 Discontinuity and linguistic theory

Is discontinuous constituency a theory-dependent notion? At first sight it might appear that way, since the notion of constituency and the role that constituent structure plays, varies from theory to theory. In the best developed grammatical theories, such as versions of Transformational Grammar, Government and Binding theory, the Barriers framework, Dependency Grammar, Lexical Functional Grammar, Augmented Phrase Structure Grammar, and Generalized Phrase Structure Grammar, the notion of phrase structure plays a central role in the formulation of the theory, and the assignment of articulated descriptions to sentences in terms of syntactic constituents is a major goal of the theory. In Head-driven Phrase Structure Grammar the situation is slightly different, in that (i) the theory focuses on the immediate-dominance aspect of constituent structure (head-daughter relations), and (ii) this aspect of constituent structure is encoded as part of the sign, the unit of linguistic information in HPSG, rather than as a separate dimension of linguistic description. In (versions of) Categorial Grammar the situation is sometimes claimed to be totally different, since CG does not pursue the explicit derivation of constituent structures, and moreover is extremely flexible in the syntactic structures the theory would allow to be assigned to sentences. Still, any CG analysis of a sentence corresponds to an implicit constituent structure being assigned to the sentence. As the contributions by Bunt and Moortgat in this volume show, this can be brought out quite clearly when CG rules are allowed to directly combine non-adjacent elements, which can be achieved by introducing discontinuity in categories.

The case of CG illustrates that in those linguistic theories that do not aim at assigning explicit syntactic descriptions to sentences, the construction of such descriptions does occur implicitly and is unavoidable, since semantic interpretation requires decisions on which elements in the sentence are
combined. This decision is largely based on syntactic evidence, including
categorial as well as word order information.

We may therefore conclude that the phenomenon of discontinuous con-
stituency is essentially theory-independent. There is some variation in the
linguistic data that are recognized as manifestations of the phenomenon,
because different theories have somewhat different views on what counts
as a constituent; still, there is a great deal of consensus about this mat-
ter across linguistic theories. As a result, the phenomenon of discontinuous
constituency provides a potentially fruitful domain for discussion across the
borders between linguistic theories. The present book aims at contributing
to this discussion.

1.2 Empirical facts about discontinuity

When we describe the syntactic organization of a sentence, we group the
words at various levels and divide the sentence into noun phrases, verb
phrases, relative clauses, adjective phrases, prepositional phrases, amount
expressions, and the like. The use of such word groups may be syntactically
motivated by the possibility to achieve a higher level of generalization by
describing syntactic structure in terms of such groups than in terms of in-
dividual words. Semantically, there is the perhaps even stronger motivation
formed by the Principle of Compositionality, which requires the decompo-
sition of sentences into meaningful parts in order to systematically derive
sentence meanings.

More often than not, the syntactically and semantically meaningful word
groups, or 'constituents', are contiguous: they consist of words standing next
to one another. But sometimes this is not the case, as in example sentence
(1):

(1) *Mary woke me up at seven-thirty.*

The verb form 'woke up' is split into two non-adjacent parts here, the par-
ticle 'up' being separated from 'woke' by 'me', which causes a discontinuity
in the verb phrase.

Exactly which discontinuities a particular grammar formalism has to
deal with, depends on the general views on constituency that the grammat-
ical theory in question takes. Theories which recognize verb phrases, for
instance, have to deal with VP discontinuities. As most theories do assume
verb phrases, it is not surprising that VP discontinuities form an impor-
tant topic of study (see e.g. the contributions to this volume by Hepple,
Hoekstra, and Reape).

Noun phrases are not so often discontinuous. But of course, relative
clause extraction leads to discontinuous NPs, and so do the VP discontinu-
ities in NP-embedded VPs, as in premodifying relative clauses in Dutch or
German. A genuine NP discontinuity is formed by Dutch determiner struc-
tures of the form *Wat voor*, as in sentence (2):

(2) *Wat heb jij voor auto gekocht?*  
(What did you car buy?, meaning What car did you buy?)

Such discontinuities are studied by Corver (this volume).

PPs and APs are mostly contiguous, but again, it is certainly not impossible for them to be discontinuous. Examples of discontinuous adjective- and adverbial phrases are presented in (3):

(3)  
a. *This is a better movie than I expected.*  
b. *Leo is harder gegaan dan ooit tevoren.*  
   (*Leo has been going faster than ever before.*)

Moreover, virtually any phrase can become discontinuous by the insertion of metacomments, such as parentheticals. Some examples:

(4)  
a. *John talked of course about politics.*  
b. *Peter bought a house on, or almost on, Angels Beach.*  
c. *Leo is going faster, I would say, than ever before.*  
d. *He invited the vice-chairman, I think it was, of the committee.*  
e. *An undergraduate student, supposedly, who had witnessed the event reported it to him.*  
f. *This designer is equally cool, you know, as Armani.*

(Examples d and e taken from Dowty, this volume.)

Even idiomatic expressions, which one may be inclined to think of as fixed, inseparable, sequences of words *par excellence*, can be discontinuous:

(5)  
a. *Will John soon kick the proverbial bucket?*  
b. *Le marchand de sable va bientôt passer.*

(See Abeillé and Schabes, this volume.)

It is thus not an exaggeration to say that discontinuities may arise almost everywhere; discontinuous constituents are by no means rare or exceptional. Still, most grammar formalisms are explicitly or implicitly based on the idea of a constituent as a continuous sequence of words, being primarily designed to describe continuous constituents and having to take recourse to special operations for dealing with discontinuities. This is because concatenation, and therefore the notion of *adjacency*, plays a central role in most grammar formalisms. The description of structures made up of non-adjacent elements therefore in general presents difficulties.
1.3 Approaches and techniques for handling discontinuity

In this book a wide variety of approaches to the study and treatment of discontinuous constituents is presented, varying from attempts to reduce the problems they pose by re-examining the notion of constituency and minimizing the articulation of syntactic description in constituent structures, to mathematical and computational techniques for dealing effectively with discontinuous constituents in linguistic descriptions and computer programs for interpreting and generating sentences with discontinuities.

Dowty in his contribution *Towards a Minimalist Theory of Syntactic Structure* takes a fundamental approach to discontinuous constituency by addressing the more general question what might count as a constituent and for what reasons. Dowty notes that

“No assumption is more fundamental in the theory (and practice) of syntax than that natural languages should always be described in terms of constituent structure, at least wherever possible.”

and goes on to note that, in spite of this,

“...syntacticians don’t usually ask questions .. as to what the nature of evidence for a constituent structure in a particular sentence in a particular language is: we just take whatever structure our favorite syntactic theory would predict as the expected one for the string of words in question - by current X-bar theory, etc. - unless and until that assumption is contradicted by some particular fact. ... I suspect syntacticians today have almost come to think of ‘the primary empirical data’ of syntactic research as phrase structure trees, so firm are our convictions as to what the right S-structure for most any given sentence is.”

Dowty advocates a more skeptical, ‘minimalist’ approach to syntactic structure which takes the default assumption that a clause or group of words is only a linear structure: a string, and that a hierarchical structure should be assigned only when necessary for getting the data right. By taking linear structure to be the norm, Dowty argues that the description of various discontinuous syntactic phenomena can be simpler. This minimal, string-like syntactic structure Dowty calls the *phenogrammatical* structure, which he contrasts with the *tectogrammatical* structure, thereby making a distinction originally due to Curry (1963). The tectogrammatical structure describes the steps by which a sentence is built up from its parts, but without regard to the form that these combinations of parts take, i.e. without information about the order of words and phrases, whether inflectional morphology marks the syntactic organization or not, and so on. The representation of the latter information is the phenogrammatical structure.
Abeillé and Schabes in their contribution use Tree Adjoining Grammar as their theoretical framework, and argue that the formalism of Lexicalized TAGs, where sets of elementary trees are associated to lexical items, defining linguistic units of extended domain of locality that have both syntactic and semantic relevance, offers a natural representation for constituents that may exhibit internal discontinuities. Such constituents follow regular syntactic TAG composition rules, but lack semantic compositionality. This representation considers noncompositional constituents as one semantic unit without having to stipulate any element to be semantically empty, and is thus kept strictly increasing.

Corver’s contribution approaches discontinuous constituency within the Barriers framework of Chomsky (1986). Specific attention is given to the discontinuity in Dutch determiner phrases that may occur in structures of the form wat voor, as in (2). It is argued that such noun phrases consist of an interrogative quasi-argument wat, that functions as the head of the NP, and a predicative PP headed by voor, which is conjoined to wat. The presence or absence of discontinuity in wat voor-phrases in various syntactic environments can be accounted for in terms of the Subjacency Condition and the Empty Category Principle on the basis of the proposed internal structure of the wat voor-phrases.

Hoekstra’s contribution, that falls within the same theoretical tradition, is concerned not just with discontinuity but more generally with phenomena of movement and binding. Two types of binding relationships are distinguished, the stronger of which is subject to a strict locality requirement. The proposed approach to binding results in analyses that, among other things, account for a recently discovered problem concerning the effect of move-alpha on the binding of negative polarity items.

The contribution by Bunt is concerned with the development of tools for formal description and effective processing of bounded discontinuities. Bunt argues first that articulate syntactic descriptions of hierarchical constituent structure do not necessarily take the form of trees. He argues, with McCawley (1982), that syntactic descriptions in general should be allowed to have crossing branches, and that there is nothing objectionable about such structures, which he calls discontinuous trees (or disco-trees, for short). A precise mathematical definition of these structures is provided. A fairly intriguing matter is the definition of a linear precedence relation among discontinuous trees in a way, useful in the formulation of grammatical rules for combining such trees. The key notion used for this purpose is the adjacency sequence, which is defined in such a way that the concatenation of two or more discontinuous trees, forming an adjacency sequence, has the effect that the elements of those trees are interwoven. Complexity-theoretical issues are briefly addressed, as are computational issues concerning their implementation in chart parsers.

The notions introduced by Bunt are theory-neutral in that they can be used in many different grammatical frameworks. In his contribution, Bunt
explains their use in phrase-structure grammars in some detail, developing
a formalism called *Discontinuous Phrase Structure Grammar* (DPSG), and
very briefly considers their possible use in Categorial Grammar. In a cate-
gorical framework, their use could be in allowing *discontinuous categories*. A
simple example would be that of a discontinuous verb phrase like ‘woke Mary
up’, here we could use a category discontinuous category \((NP\$) [\!NP\!] /PART\)
for the verb (‘woke’), which concatenates to the right in an adjacency se-
quence \((NP\$) [\!NP\!] /PART + NP + PART \Rightarrow (NP\$)/NP\) as follows:

\[(6) (NP\$) [\!NP\!] /PART + NP + PART \Rightarrow (NP\$)/NP\]

The part \([\!NP\!]\) in the discontinuous category indicates that a noun phrase
(‘Mary’) may separate a VP-final particle (‘up’) from the verbal stem (‘woke’);
the verb and the particle form a discontinuous functor which applies to the
separating NP as its right-hand side argument.

Moortgat, in his contribution, goes at great length in studying the conse-
quences and possibilities of discontinuous categories. He considers two discon-
tinuous type constructors, intended to capture extraction and infixation
operations at a level suitable for linguistic generalization. In the first case,
the operator constructs a discontinuous category of which the two constituent
parts together form a discontinuous functor, applicable to the intervening
material; in the other case (infixation) the constructor constructs a discon-
tinuous argument for the functor corresponding to the intervening material.
These operators had been introduced before in Moortgat (1988) in an at-
tempt to accommodate Bach’s (1981; 1984) work on discontinuity within a
categorial type logic approach. In his contribution, Moortgat improves on
his earlier proposal by presenting a complete logic for extraction and infix-
ation type constructors in terms of the sign-based approach to categorial
deduction.

Sanfilippo in his contribution develops the idea that the relevant con-
straints on discontinuous verb-object dependencies, arising in Italian sen-
tences like ‘A Maria, Carlo non gli ha ancora scritto’ (‘To Mary, Carlo has
not written to-him yet’), can be expressed in terms of access to the thematic
domain of a sentence. It is shown how a strong generalization expressed in
these terms can be given a precise syntactic, semantic and computational
interpretation within the framework of Unification Categorial Grammar aug-
mented with a neo-Davidsonian verb semantics and predicate-argument as-
sociation. The resulting approach can easily be extended to also provide an
empirically motivated account of dislocated subjects, and is thus interesting
for the treatment of discontinuous subject-VP dependencies as well.

Hepple’s contribution is also in the framework of Categorial Grammar.
He presents a novel view of the relation between word order and subcat-
egorization, according to which the canonical word order in any language
arises through the interaction of three factors: (i) the ‘underlying’ order of
subcategorization by the verb for its complements corresponding to oblique-
ness; (ii) the direction (left or right) of subcategorization by the verb of each complement; (iii) a lexical process of 'verb displacement', causing the verb to be displaced from its position relative to its complements according to the underlying subcategorization.

Pollard's contribution is concerned with a detailed study of verb-second and raising phenomena in the framework of Head-driven Phrase Structure Grammar. In Government-Binding theory and associated frameworks it has become standard to regard occurrence in second position of the finite verb in declarative sentences (in English, Dutch, German, and many other languages) as a case of head movement. The finite verb or auxiliary, on this account, undergoes movement into the COMP position, and some phrasal constituent moves to the left of it. The apparent discontinuity between the finite auxiliary and its VP sister in such sentences might be thought to be problematic for nontransformational theories. Pollard discusses this issue in detail for verb clusters in German, and gives an account of the possibilities and constraints to front parts of such clusters. A satisfactory analysis of these phenomena is proposed within the HPSG framework, with one issue not entirely resolved: the resulting set of combination schemata allow different parses for certain sentences, with no demonstrable linguistic (e.g., pragmatic) differences between the alternative analyses. Exactly how this problem of spurious ambiguity can be resolved is not quite clear.

Reape's contribution is also in the framework of HPSG. He argues that surface syntax should be rejected as the basis of word order. Instead, he claims that word order is determined within locally definable 'word order domains', where nonlexical word order domains are composed from smaller word order domains, and where the word order domain of a daughter may be the same as that of the mother. This last stipulation is the basis for allowing discontinuities, as it means, informally, that the elements of the daughters's domain can be interleaved with those of the mother's domain. Technically, Reape introduces an operation called sequence union to be able to describe an 'interleaving concatenation' of discontinuous constituents. The sequence union operator applied to two strings $A$ and $B$ can produce any sequence of elements from $A$ and $B$ which conserves the relative order among the $A$-elements and that among the $B$-elements. This concept is similar to Bunt's notion of concatenating adjacency sequences, the latter being slightly more constraining.

Aarts in his contribution introduces a variant of Bunt's DPSG framework, which he calls Discontinuous Rewrite Grammar. He shows how the formalism can describe both bounded and unbounded dependencies in terms of discontinuous trees with crossing branches, and pays particular attention to the design of an elegant and efficient chart parser for this formalism.

De Smedt and Kempen describe a grammar formalism designed for incremental generation of sentences, called Segment Grammar. This formalism is somewhat reminiscent of Tree-Adjoining Grammar, but differs in that sister nodes can be incrementally added to an existing (partial) structure. The for-
Malism distinguishes two levels of syntactic description: *f*-structures, which are unordered functional structures, and *c*-structures, which represent left-to-right order of constituents. True discontinuities are viewed as differences between immediate dominance relations in *c*-structures and those in corresponding *f*-structures. Constructions that are treated in this way include clause union (including Dutch cross-serial dependencies), right dislocation, and fronting.

Their contribution is concerned with discontinuous constituency not only from a structural viewpoint but also from a processing point of view. They note that several kinds of discontinuities seem to offer advantages for an incremental generation strategy; this is especially true of optional dislocations. Fronting of focused constituents is natural if we assume prominent concepts to be passed on to the grammatical encoder earlier than other parts of the semantic input. Similarly, right dislocation allows the generator to utter constituents which are ready and postpone uttering more complex ('heavy') ones, which are still being processed semantically. In addition, right dislocation allows the incorporation of new semantic input as afterthoughts.

Saint-Dizier's contribution is devoted to new techniques for expressing constraints and relations on discontinuous elements in a structure in a computationally attractive way. To this end, an extension to the logic programming language Prolog is presented, called *Dislog*, especially designed for logic programming with discontinuities. A Dislog clause is a set of definite clauses, possibly sharing common variables, which must co-occur in a parse (or proof) tree. The notion of discontinuity is expressed by the fact that there are no *a priori* restrictions on the locations of these definite clauses in the parse (proof) tree. Saint-Dizier explains the concepts of Dislog and shows how Dislog can be applied to the parsing of discontinuous constituents in natural language.

References


Chapter 2

Formal tools for describing and processing discontinuous constituency structure
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.
2.1 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 ordered sets of words\(^1\) 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.

\[ S \]
\[
\begin{array}{c}
S \\
NP & V' \\
V & PP \\
John talked of course about politics
\end{array}
\]

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

\(^1\) Or rather, multiset{s} of words, i.e. unordered collections, possibly with duplicates.
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 continuous. 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).

\[
(2) \quad \text{persuade to go} \quad - \quad \text{Vt/ViP} + \text{ViP} \\
\quad \text{consider competent} \quad - \quad \text{Vt/AdjP} + \text{AdjP}
\]

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).

\[
(3) \quad \text{RWRAP}(a,b) = \begin{cases} 
\text{a + b} & \text{if a is simple} \\
\text{X + b + W} & \text{if a has the form } [x P X W] 
\end{cases}
\]

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.\(^2\)

\(^2\)See Vogel & Bunt (1992) for a discussion of the relation between Right-Wrap and
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 strong 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) \[ VP \Rightarrow V_i \ NP \ TVP[SL : V_i] \]

Promotion of the SLASH feature is used to account for the discontinuity created by relative clause extraposition as well as that represented by cross-
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 variety 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 neighbourhood) 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.

### 2.2 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.

![Discontinuous Tree](image)
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 \(y\):

\[
\text{(8) For any two nodes } x \text{ and } y \text{ in a tree, } x < y \text{ iff for all nodes } u \text{ and } v, \text{ if } x \text{ dominates } u \text{ and } y \text{ dominates } v, \text{ then } u < v.
\]

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

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

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:

\[
\text{(10) For any two nodes } x \text{ and } y \text{ in a tree, if } x \ D' \ y \text{ then neither } x < y \text{ nor } y < x.
\]

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:

\[
\text{(11) Let } N \text{ be a non-empty set whose elements are called NODES.}
\]

\[
\text{(i) If } t \in N, \text{ then } <t,[]> \text{ is a tree, where } [] \text{ represents the empty list;}
\]
(ii) If $X_1, X_2, \ldots, X_k$ are disjoint trees and $a \in \mathbb{N}$, then $<a, [X_1, X_2, \ldots, X_k]>$ is a tree.

Instead of `$<a, [X_1, X_2, \ldots, X_k]>$', we shall also write more simply: `$(X_1, X_2, \ldots, X_k)$'.

(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) S
   /\  \\
  P  Q
 / \ \\
/   /\  \\
 a   b  c  d
```

'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 $Q$: 
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 ∈ N, then the pair <t, ]> is a subdiscotree; such a subdiscotree is called ATOMIC;

(ii) If x ∈ N and X₁, .., Xₖ are subdiscotrees that do not share any
  subdiscotrees⁴ then the pair <x, [X₁, Y₂, ..., Yₖ, Xₖ]> is a sub-
  discotree, where Yᵢ = Xᵢ or Yᵢ a sub-list of {X₂, ...,Xₖ₋₁}.

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

Instead of `x(X₁, .., Xₖ)`, we also write, more simply: `x(X₁, .., Xₖ)'. The idea is that a subdiscotree consists of a top node
x and a sequence of nodes dominated by x (daughter nodes), possibly in-
terrupted by lists [Xᵢ, Xᵢ, ..] 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 repre-
sentation 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 subdis-
cotrees as follows:

(18) Direct dominance (D):

³Two subdiscotrees X and Y share a subdiscotree T if both X and Y contain a node
  dominating T.
(i) in an atomic subdiscotree \(<x, [\ ]>\), there is no node that is dominated by node \(x\);
(ii) in a subdiscotree \(<x, [X_1, Y_1, \ldots, Y_j, X_k]>\) 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 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.

2.2.1 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 \(A(X_1, \ldots, X_n)\), \(X_i < X_j\) iff \(i < j\).
(ii) If P, Q, x and y are nodes in a tree such that P < Q, P D x, and Q D y, then x < 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).

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_j, ..] .., 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.

### 2.3 Discotrees in grammar rules

#### 2.3.1 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):
To generate this structure, we need rules like the following, where square brackets indicate internal context elements:

\[
\begin{align*}
(25) \quad & \text{VP} \rightarrow V + \text{NP} \\
& V \rightarrow \text{VS} + [\text{DET}] + [N] + \text{PART} \\
& \text{NP} \rightarrow \text{DET} + N \\
& \text{VS} \rightarrow \text{wake} \\
& \text{DET} \rightarrow \text{your} \\
& N \rightarrow \text{friend} \\
& \text{PART} \rightarrow \text{up}
\end{align*}
\]

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):

(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 \( <x, []> \), then \( Lm(X) = x \);
(ii) else X is a subdiscotree of the form \( <x, [X_1, ..., [..],.., X_n]> \) and \( Lm(X) = X_1 \).

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) \( Lm(x) < Lm(y) \);
(ii) for every node z such that \( Lm(x) < z < Lm(y) \): 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.

2.3.2 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.
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.

2.3.3 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:
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).

2.3.4 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:

\[
(39) \quad R \rightarrow A + [B] + [C] + D
\]

\[
(40) \quad \text{ID: } R \rightarrow \{A, D\} \\
\text{CD: } \{B, C\} \\
\text{LP: } A < D; B < C.
\]

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:

\[
(41) \quad \text{ID: } R \rightarrow \{A, D\} \\
\text{CD: } \{B, C\} \\
\text{LP: } A < D.
\]
The LP rule 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 disctrees 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,
COMPONENT-DAUGHTER, ADJUNCT-DAUGHTER, FILLER-DAUGHTER are
used to represent mother-daughter relations with certain functional char-
acteristics. 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)disctrees, 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 propa-
gate according to certain general principles, like the Subcat Principle. Ex-
pectations concerning complement daughters are expressed in the values of
the SUBCAT attribute, those concerning filler daughters in the SLASH at-
ttribute, and those concerning head-adjunct daughter combinations either in
the ADJUNCTS feature (Pollard & Sag, 1987) of the head daughter 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.

2.4 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, it has properly less power than the Mildly Context-

4The restriction considered in these studies is that any node c, which occurs as a
c context daughter of a node t, must also be directly dominated by some node s, distinct
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, 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). 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 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).

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.


6For developments in this pragmatic theory since the TENDUM project see Bunt (1990; 1994).
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 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

European Chapter of the ACL, Copenhagen 1987, 203-211.


<table>
<thead>
<tr>
<th>No</th>
<th>Author</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>H.C. Bunt</td>
<td>On-line Interpretation in Speech Understanding and Dialogue Systems</td>
</tr>
<tr>
<td>2</td>
<td>P.A. Flach</td>
<td>Concept Learning from Examples Theoretical Foundations</td>
</tr>
<tr>
<td>3</td>
<td>O. De Troyer</td>
<td>RIDL*: A Tool for the Computer-Assisted Engineering of Large Databases in the Presence of Integrity Constraints</td>
</tr>
<tr>
<td>4</td>
<td>M. Kammler and E. Thijsse</td>
<td>Something you might want to know about &quot;wanting to know&quot;</td>
</tr>
<tr>
<td>5</td>
<td>H.C. Bunt</td>
<td>A Model-theoretic Approach to Multi-Database Knowledge Representation</td>
</tr>
<tr>
<td>6</td>
<td>E.J. v.d. Linden</td>
<td>Lambek theorem proving and feature unification</td>
</tr>
<tr>
<td>7</td>
<td>H.C. Bunt</td>
<td>DPSG and its use in sentence generation from meaning representations</td>
</tr>
<tr>
<td>8</td>
<td>R. Berndsen and H. Daniels</td>
<td>Qualitative Economics in Prolog</td>
</tr>
<tr>
<td>9</td>
<td>P.A. Flach</td>
<td>A simple concept learner and its implementation</td>
</tr>
<tr>
<td>10</td>
<td>P.A. Flach</td>
<td>Second-order inductive learning</td>
</tr>
<tr>
<td>11</td>
<td>E. Thijsse</td>
<td>Partial logic and modal logic: a systematic survey</td>
</tr>
<tr>
<td>12</td>
<td>F. Dols</td>
<td>The Representation of Definite Description</td>
</tr>
<tr>
<td>13</td>
<td>R.J. Beun</td>
<td>The recognition of Declarative Questions in Information Dialogues</td>
</tr>
<tr>
<td>14</td>
<td>H.C. Bunt</td>
<td>Language Understanding by Computer: Developments on the Theoretical Side</td>
</tr>
<tr>
<td>15</td>
<td>H.C. Bunt</td>
<td>DIT Dynamic Interpretation in Text and dialogue</td>
</tr>
<tr>
<td>16</td>
<td>R. Ahn and H.P. Kolb</td>
<td>Discourse Representation meets Constructive Mathematics</td>
</tr>
<tr>
<td>No</td>
<td>Author</td>
<td>Title</td>
</tr>
<tr>
<td>----</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>17</td>
<td>G. Minnen and E.J. v.d. Linden</td>
<td>Algorithmen for generation in lambek theorem proving</td>
</tr>
<tr>
<td>18</td>
<td>H.C. Bunt</td>
<td>DPSG and its use in parsing</td>
</tr>
<tr>
<td>19</td>
<td>H.P. Kolb and C. Thiersch</td>
<td>Levels and Empty? Categories in a Principles and Parameters Approach to Parsing</td>
</tr>
<tr>
<td>20</td>
<td>H.C. Bunt</td>
<td>Modular Incremental Modelling Belief and Intention</td>
</tr>
<tr>
<td>21</td>
<td>F. Dols</td>
<td>Compositional Dialogue Referents in Prase Structure Grammar</td>
</tr>
<tr>
<td>22</td>
<td>F. Dols</td>
<td>Pragmatics of Postdeterminers, Non-restrictive Modifiers and WH-phrases</td>
</tr>
<tr>
<td>23</td>
<td>P.A. Flach</td>
<td>Inductive characterisation of database relations</td>
</tr>
<tr>
<td>24</td>
<td>E. Thijsse</td>
<td>Definability in partial logic: the propositional part</td>
</tr>
<tr>
<td>25</td>
<td>H. Weigand</td>
<td>Modelling Documents</td>
</tr>
<tr>
<td>26</td>
<td>O. De Troyer</td>
<td>Object Oriented methods in data engineering</td>
</tr>
<tr>
<td>27</td>
<td>O. De Troyer</td>
<td>The O-O Binary Relationship Model</td>
</tr>
<tr>
<td>28</td>
<td>E. Thijsse</td>
<td>On total awareness logics</td>
</tr>
<tr>
<td>29</td>
<td>E. Aarts</td>
<td>Recognition for Acyclic Context Sensitive Grammars is NP-complete</td>
</tr>
<tr>
<td>30</td>
<td>P.A. Flach</td>
<td>The role of explanations in inductive learning</td>
</tr>
<tr>
<td>31</td>
<td>W. Daelemans, K. De Smedt and J. de Graaf</td>
<td>Default inheritance in an object-oriented representation of linguistic categories</td>
</tr>
<tr>
<td>32</td>
<td>E. Bertino and H. Weigand</td>
<td>An Approach to Authorization Modeling in Object-Oriented Database Systems</td>
</tr>
<tr>
<td>33</td>
<td>D.M.W. Powers</td>
<td>Multi-Modal Modelling with Multi-Module Mechanisms: Autonomy in a Computational Model of Language</td>
</tr>
<tr>
<td>No</td>
<td>Author</td>
<td>Title</td>
</tr>
<tr>
<td>----</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>34</td>
<td>R. Muskens</td>
<td>Anaphora and the Logic of Change*</td>
</tr>
<tr>
<td>35</td>
<td>R. Muskens</td>
<td>Tense and the Logic of Change</td>
</tr>
<tr>
<td>36</td>
<td>E.J. v.d. Linden</td>
<td>Incremental Processing and the Hierarchical Lexicon</td>
</tr>
<tr>
<td>37</td>
<td>E.J. v.d. Linden</td>
<td>Idioms, non-literal language and knowledge representation 1</td>
</tr>
<tr>
<td>38</td>
<td>W. Daelemans and A. v.d. Bosch</td>
<td>Generalization Performance of Backpropagation Learning on a Syllabification Task</td>
</tr>
<tr>
<td>39</td>
<td>H. Paijmans</td>
<td>Comparing IR-Systems: CLARIT and TOPIC</td>
</tr>
<tr>
<td>40</td>
<td>R. Muskens</td>
<td>Logical Omniscience and Classical Logic</td>
</tr>
<tr>
<td>41</td>
<td>P. Flach</td>
<td>A model of induction</td>
</tr>
<tr>
<td>42</td>
<td>A. v.d. Bosch and W. Daelemans</td>
<td>Data-oriented Methods for Grapheme-to-Phoneme Conversion</td>
</tr>
<tr>
<td>43</td>
<td>W. Daelemans, S. Gillis, G. Durieux and A. van den Bosch</td>
<td>Learnability and Markedness in Data-Driven Acquisition of Stress</td>
</tr>
<tr>
<td>44</td>
<td>J. Heemskerk</td>
<td>A Probabilistic Context-free Grammar for Disambiguation in Morphological Parsing</td>
</tr>
<tr>
<td>45</td>
<td>J. Heemskerk and A. Nunn</td>
<td>Dutch letter-to-sound conversion, using a morpheme lexicon and linguistic rules</td>
</tr>
<tr>
<td>46</td>
<td>A. HH. Ngu, R. Meersman and H. Weigand</td>
<td>Specification and verification of communication constraints for interoperable transactions</td>
</tr>
<tr>
<td>47</td>
<td>J. Jaspars and E. Thijsse</td>
<td>Fundamentals of Partial Modal Logic</td>
</tr>
<tr>
<td>48</td>
<td>E. Krahmer</td>
<td>Partial Dynamic Predicate Logic</td>
</tr>
<tr>
<td>49</td>
<td>W. Daelemans</td>
<td>Memory-Based Lexical Acquisition and Processing</td>
</tr>
<tr>
<td>50</td>
<td>G. Rentier</td>
<td>A Lexicalist Approach to Dutch Cross Serial Dependencies</td>
</tr>
<tr>
<td>51</td>
<td>R. Muskens</td>
<td>Categorial Grammar and Discourse Representation Theory</td>
</tr>
<tr>
<td>No</td>
<td>Author</td>
<td>Title</td>
</tr>
<tr>
<td>----</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>52</td>
<td>M. Rats</td>
<td>Topic-Comment Structures in Information Dialogues</td>
</tr>
<tr>
<td>53</td>
<td>G. Rentier</td>
<td>Extraction in Dutch with Lexical Rules</td>
</tr>
<tr>
<td>54</td>
<td>H. Bunt</td>
<td>Dialogue control functions and interaction design</td>
</tr>
<tr>
<td>55</td>
<td>A. van Horck</td>
<td>Document Servers and the Electronic Distribution of Grey Literature</td>
</tr>
<tr>
<td>56</td>
<td>H. Bunt</td>
<td>Discontinuous Constituency: Introduction and Formal Tools for Description and Processing</td>
</tr>
</tbody>
</table>