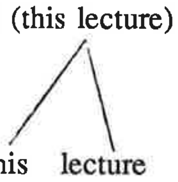
EQUIVALENCES BETWEEN CONSTITUENCY AND DEPENDENCY

Dan Maxwell
BSO/Language Technology
Postbus 543
3740 AM Baarn
maxwell@ltb.bso.nl

1. *Introduction*

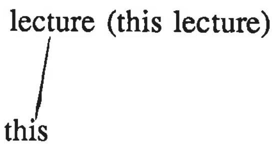
Most work in generative grammar over the past several decades has made use of a concept known as **Constituency**. This can be informally defined as the way words combine to form larger units. The tree in (1), for example, shows how the words *this* and *lecture* combine to form the larger unit *this lecture*. The two individual words occupy what are known as lexical nodes at the bottom of the tree. The result of this combination is provided at a separate node, known as a phrasal node.

(1)



During roughly the same period a different concept known as **Dependency** was also developed (Tesniére 1959), though by a much smaller group of linguists. An informal definition of this concept might be that one word A is dependent on another word B, if B is a necessary prerequisite for the occurrence of A in this particular usage. The tree in (2) shows that *this* depends on *lecture*. The words in parentheses show the result of combining these words.

(2)



Dependency trees, unlike constituency trees, do not make a separate node available to show the result of this combination. Indeed, work in dependency grammar has as far as I know concentrated on the question of how trees should look in order to represent the relationship between words in the sentence, not how these words are generated and not how the meanings of these words combine with each other to create the meaning of such larger units. At least some of the points I make in this paper are not entirely new (See Schubert 1987 for an overview of work within the dependency tradition). But even within this work, little research has been done on dependency structures in the light of proposals presented in more recent generative frameworks (Hudson 1984 and Mel'čuk 1987 are to some extent exceptions). I will suggest on the basis of my findings here that this connection is worth more attention, since it appears that dependency grammar can be fruitfully combined with at least some generative frameworks.

2. *Sibling as Head, Parent as Governor*

The tradition making use of constituent structure has clearly been dominant in linguistic research, but starting with the development of X-bar theory (Jackendoff 1977) work in generative grammar frameworks has been strongly influenced by the dependency tradition, although this influence is seldom acknowledged in the literature. Thanks to Jackendoff's work, it is clear that constituency trees also express dependency. e.g. in (1), the lexical N *lecture* is the head of the entire NP and has the determiner *this* as a dependent. More generally, we can say that lexical "heads" in constituency trees have their siblings as dependents. The difference between constituency and dependency trees is that in the latter the governor of any node is its parent rather than the sibling node with the same category as the parent or the one specially designated as head.

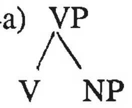
And we have seen in (2) how a dependency tree can be interpreted as a representation of how a dependent combines with a parent word in a dependency tree to form a larger unit. Under this interpretation, dependency trees express constituency. As mentioned before, it is not customary to write the result of this combination in dependency trees. But if some kind of dependency framework were to be combined with a formal semantics in the tradition started by Montague's work, then this would become a necessary step in any case. And the distinction made in HPSG between **local** and **nonlocal** components of the feature structures at a given node provide a further basis for such a step, since the nonlocal component of a given node presumably contains, among other things, the meaning of the subtree dominated by that node.

In (3-5), we see a series of trees represented both as constitituent trees and dependency trees. This time I have also added categories. The trees in (3a) and (4a) would be generated in any generative framework, as far as I know. The tree in (5a) has the hierarchical structure generated in GPSG (Gazdar et al., 1985) and HPSG (Pollard and Sag 1987), though not in GB. The words in dependency trees are usually written at the nodes themselves, since as discussed in section 4, the categories and the dependency relationships do not themselves uniquely determine the linear order of these words.

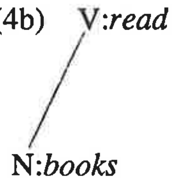
(3a) PP
/\
P NP

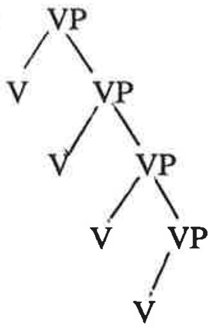
on linguistics

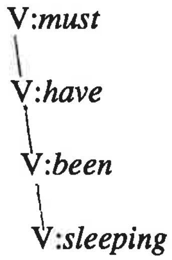
(3b) P:*on*
/
N:*linguistics*

(4a) 

read books

(4b) 

(5a) 

(5b) 

must have been sleeping

These trees make it clear that in a great many cases the structure of a dependency tree is exactly parallel to its corresponding constituency tree. The difference is that in constituency trees separate nodes are introduced for the lexical head and the combination of this head with its complement, whereas in dependency trees these two different kinds of information are combined within a single node.[1]

3. **Differences between typical constituency trees and dependency trees**

While discussing a striking similarity between some kinds of constituency trees and dependency trees, I also mentioned a few differences. Here they are again with one new one (8) added:

---

[1] There are some constructions for which this correlation does not hold:

(i) complementizers are normally assumed to govern dependent clauses in dependency grammar, though they are not usually considered to be lexical heads of clauses in constituency trees.

(ii) a finite VP is a **phrasal head** of a clause, but has no corresponding node at all in dependency grammar, since phrasal nodes are generally not part of such frameworks.

(iii) in GPSG, at least, additional VPs are introduced iteratively as parents of adjuncts. These have no correlate in dependency grammar for the reason mentioned in (ii).

(iv) "filler" constructions, used in GPSG and HPSG to deal with long distance dependencies, have an S dominating the filler and another S node, which is accordingly the (phrasal) head among this sibling pair. Although it is perhaps not entirely clear how long distance dependencies should be treated in dependency grammar, none of the possibilities correlate with the constituency analyses the way (3a-5a) correlate with (3b-5b).

(6) There are no abstract phrasal categories in dependency trees.

(7) Syntactic categories are expressed both by a lexical head and their phrasal extensions in constituency trees.

(8) The linear order of words is in general visually expressed in constituency t rees but not in dependency trees..

A consequence of (7) is that
some principle is necessary to enable sharing of information between a lexical head and its phrasal extension(s). This is demonstrated in the following example from German. The phrase structure or "immediate dominance" rule in (9a), as well as its dependency based correlate in (9b), is presented not only to show another correlation between constituency and dependency frameworks, but to show how the feature **accusative** might be introduced at an NP node. This feature has to be introduced at this node because of the dependency of this case on the lexical head of the rule, which must be a transitive verb.

(9a)

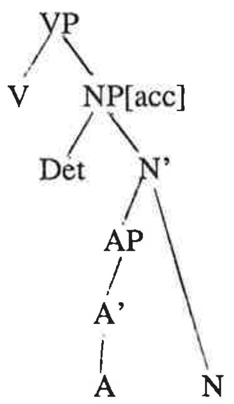
VP-->V NP[acc]

(9b)

V-->N[acc]

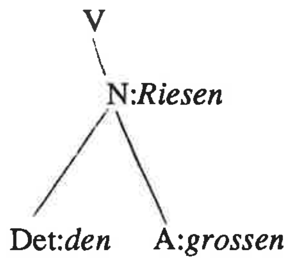
den grossen Riesen
the(acc) big(acc) giant(acc)

(10a)



den grossen Riesen

(10b)



But of course the accusative case is present phonetically only on the lexical extension of the NP. In the same way, once the feature **accusative** has been instantiated by some sort of agreement principle (the *Control-Agreement Principle* in GPSG or *structure sharing* in HPSG) on the AP, some way has to be provided of making it appear on the lexical Adjective as well.

In other words, some principle is necessary to ensure that certain features such as case are shared between lexical heads and their phrasal extensions. Such principles have been proposed in several frameworks. They differ merely in name and the way they interact with other properties of the framework concerned. They include the following:

Projection Principle (GB)
Head Feature Convention (GPSG)
Head Feature Principle (HPSG)

Such a principle is evidently superfluous in dependency grammar, since such features are introduced directly on the single node which serves to encorporate both lexical information and whatever phrasal information is relevant.

4. **Linearization**

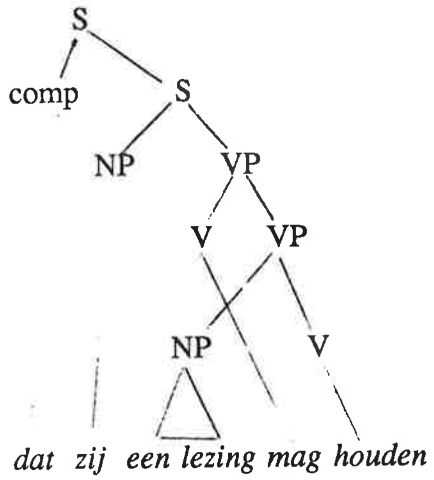
I now turn to the question of linear order of words in a sentence. Although typical constituency trees present the words of a sentence in the linear order used in actual speech (cf. (8), above), there are some kinds of sentences in which it is difficult to maintain this property without modifying the assumption that the siblings of the lexical head are its arguments.

If constituency trees continue to express dependency relationships as sibling relationships between lexical heads and complements, then in cases involving **discontinuity**, they, like dependency trees, fail to visually express linear order.

Dutch is a language in which discontinuities are triggered by a whole class of verbs rather than only isolated lexical items (Bresnan et al. 1982). The following example is such a sentence. If the NP is assumed to be a sibling of its lexical head, the verb *houden*, then the only way this tree can visually express the linear order of these constituents is by permitting crossing branches, as in (11):

dat zij een lezing mag houden
that she a talk may hold
"that she may give a talk"

(11)



Trees with crossing branches have not found wide acceptance as a formal descriptive device in generative grammar. The formal definition of "tree structure" as given in textbooks on mathematical linguistics (Wall 1970) explicitly excludes this possibility, since this would extend the power of any framework in ways difficult to define.

An alternative to the idea of expressing linear order of the words in a sentence by their order at the bottom of the tree is to do this by numbers. Such an approach presupposes some sort of procedure for correctly assigning the numbers. Such a procedure is generally available in the system of ID rules and LP statements used in GPSG and HPSG, although this procedure currently has no fully satisfactory way of dealing with discontinuities. [2]

Such a system can be adapted to a dependency grammar as follows:

LP statements in dependency grammar express relative order between **Governor** and **Dependent.**

A precise interpretation of LP statements within a dependency framework is the following:

> for any two nodes X and Y such that X governs Y, the LP statement X < Y means that X and any dependents other than Y must precede Y, i.e. be assigned a lower number than Y, and **all** of its dependents. If all the {X,Y} pairs in the tree obey all of the LP statements, then the tree is "LP-acceptable" (this term is

[2] Reape 1990 offers an extension of HPSG designed to deal with discontinuities in the way just outlined.

borrowed from Gazdar et al. 1985:99).

Such LP statements express constraints on what might be called "G-D (Governor-Dependent) linearization." These must be supplemented by rules (or principles) expressing the relative order of siblings in just those subtrees in which more than one offspring precedes or follows the parent. An example in English would be sentences in which the verb governs both an object and an prepositional complement, such as *I gave the book to Tom*, since both of these complements follow the governing verb.

In HPSG, a principle called the "Obliqueness Hierarchy", based roughly on the NP Accessibility Hierarchy proposed in Keenan & Comrie 1977 for entirely different reasons, is used to deal with dependents which are also complements. Pollard and Sag assume that this principle is specific to English, but in fact it seems to have some validity - in many cases perhaps only as the "default principle" - to a wide variety of languages. Such a principle could perhaps be extended to adjuncts as well and in any case seems to be applicable without change to any dependency-based version of HPSG. But in the example below, no such principle is in fact necessary, since the ordering of the only pair of siblings in the tree is uniquely determined as a by-product of G-D linearization.

In dependency frameworks, but not constituency frameworks, sentences with different word orders, but the same dependency relationships, can have trees which are identical except for the numbers determining linear order, as the following example based on Dutch and English shows. I give in addition to the trees the linearization of the tree as it develops step by step from the top down. To the right of the trees are the number(s) of the LP statement(s) which is (are) applicable at that level of the derivation.

(12a) *zij heeft een lezing gehouden*
she has a talk held
"she has given a talk"

heeft:V[fin],2 (heeft)

zij:N,1 *gehouden*:V[psp],5 (zij heeft gehouden)[(i), (ii)]

lezing:N,4 (zij heeft lezing gehouden) [(i)]

een:N,3 (zij heeft een lezing gehouden)[(i)]

(i) G > D
(ii) [fin] < [-fin]

where '< (>)' means 'has a lower (higher) number than'

(12b) *she has given a talk*

```
         has:V[fin],2           (has)
        /   \
she:N,1   given:V[psp],3 (she has given)        [(ii),(i)]
             \
             talk:N,5   (she has given talk)        [(i)]
               \
               a:N,4    (she has given a talk)      [(iii)]
```

(i) G < D
(ii) [fin] > N
(iii) N > Det

These LP rules are chosen largely for the sake of clarity in these examples, although consideration of a larger amount of data would very likely justify some kind of adjustment.

The interaction among the individual LP statements in either set is based on the hierarchy of specificity proposed (for a somewhat different purpose) in Schubert 1987:162ff. The first rule in each case is the default rule by virtue of the fact that it is less specific than the other(s); G is a metavariable for any governor and D a metavariable for any dependent. The default rules apply to a given {G,D} pair whenever none of the more specific rules meets the conditions for application to this pair.

If we interpret LP statements procedurally, it is in fact necessary to interpret LP rules as requiring adjacency between the governor and dependent at the time of application of the rule. Otherwise, it would be possible on the basis of LP (iii), for example, to place the determiner *a* anywhere in the sentence that precedes its governor *talk*.

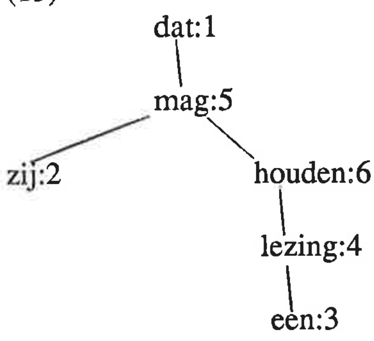
But at the beginning of the discussion of LP statements, I gave a declarative interpretation which requires not only the dependent word itself, but also all its descendants to have a specific precedence relationship with respect to its governor. This is in itself sufficient to exclude any of the undesired results. If *a* were to precede *gave*, for example, this would violate LP (i), since this requires not only any dependent of *gave*, but also any descendants of such a dependent to follow it.

It is interesting to speculate how the case of discontinuity shown in (11) might be dealt with in a dependency-based framework. The first point to be made in this regard is the fact that there is no difficulty in generating a tree for such sentences; crossing branches do not occur, since the form of the tree is not constrained by the order of the words in the sentence. We shall subsequently see, however, that the converse of this relationship is valid: the form of the tree constrains the order of words.

The tree is shown in (13). One additional LP rule (iii) is necessary to deal with the relative order of the complementizer *dat* and its dependent. The numbering of the other LP rules refers to the numbering in the previous example.

(iii) comp < V

(13)



(dat)

(dat mag) [iii]

(dat zij mag houden) [(ii),(i)]

(dat zij lezing mag _ houden) [(i)]

(dat zij een lezing mag _ houden) [(i)]

In a dependency framework, the problem of discontinuity is reduced to a problem of linearization. On the basis of LP statements as defined above, it is impossible to derive the sequence of numbers on the above tree. In the derivation of linear order to the right, the dash indicates where the dependent ought to occur according to the LP rule applied. The order which actually occurs ought to be excluded by LP (ii), since this excludes any order in which any descendant of *mag* precedes it, and *een lezing* is such a descendant.

The problem with this tree is apparently that the dependent *een lezing* is ordered not with respect to its governor *houden*, as we would expect, but rather with respect to the parent *mag* of this governor. It seems to be necessary to introduce something in the tree which makes such a procedure possible in just this sort of syntactic environment. But I will not attempt to show how this could be done here.

5. Conclusion

I have argued that the fundamental differences between constituency and dependency are actually less significant than generally seems to be supposed. Three fundamental developments in generative grammar - X-bar theory, the conception of a node on a tree as a complex of features with its own internal structure, and the ID/LP framework - have made comparison between consituency trees and dependency trees more feasible than previously. It even appears possible to create a form of generative grammar based on dependency trees. Since such trees have many fewer nodes than their corresponding constituency trees, I would expect such a framework to be both easier to deal with computationally and also easier to teach and learn than a corresponding framework based on constituency.

References

Bresnan, Joan, Ron Kaplan, Stanley Peters, Annie Zaenen. 1982. Cross-Serial dependencies in Dutch. *Linguistics Inquiry*13: 613-35.

Gazdar, Gerald, Ewan Klein, Geoffrey Pullum, Ivan Sag. 1985. *Generalized Phrase Structure Grammar*. Oxford: Basil Blackwell.

Hudson, Richard. 1984. *Word Grammar*. Oxford: Basil Blackwell.

Jackendoff, Ray. 1977. *X-bar Syntax: A Study of Phrase Structure*. Cambridge, Massachusetts: MIT Press.

Keenan, Edward & Bernard Comrie. 1977. Universal grammar and noun phrase accessibility. *Linguistic Inquiry* 8.1: 63-99.

Mel'čuk, Ivan. 1986. *Dependency Grammar: Theory and Practice*. Albany: State University of New York Press.

Pollard, Carl, Ivan Sag. 1987. *Information-based Syntax and Semantics*. Stanford: CSLI.

Reape, Mike. 1990. Getting things in order. 125-139 in the proceedings of symposium on Discontinuity. Tilburg: Institute for Language Technology and Artificial Intelligence.

Schubert, Klaus. 1987 *Metataxis: Contrastive Dependency Syntax for Machine Translation*. Dordrecht: Foris Publications.

Tesnière, Lucienne. 1959. *Éléments de syntaxe structurale*. Paris: Klinksieck [2.ed., 4. print. 1982].

Wall, Robert. 1972. *Introduction to Mathematical Linguistics*. Englewood Cliffs, New Jersey: Prentice Hall.