Discontinuous phrases in dependency grammar[*]

RICHARD HUDSON

1 Introduction

The notion 'phrase' is recognised by every syntactic theory, as a group of words which behave syntactically (and semantically) as a single unit. In straightforward cases the words in a phrase are next to each other, without interruptions from outside the phrase, and discontinuous phrases are ungrammatical. For example, the phrase *green peas* is continuous, and grammatical, in example a) but not in b).

(1) a He lives on green peas.
b *He lives green on peas.

On the other hand, some phrases manage to be both discontinuous and grammatical at the same time. For example, extraction regularly produces discontinuous phrases like *what ... he wanted* in b) below, and subject-raising gives phrases like *it ... raining* in a). The following list illustrates four of the better-understood and more important discontinuous constructions of English. These are the patterns I shall focus on in this paper, though there are others.[1]

(2) a *It* keeps *raining*. (raising)
b *What* did he say *he wanted?* (extraction)
c *Mary* is easy to *please.* (tough-movement)
d *Students* always do well *who take syntax.* (extraposition)

The evidence for these discontinuous phrases is quite compelling, and not tied to any particular theory; in each case some rules apply to the underlined words in just the

[*]The ideas in this paper have been growing over the last two years, through discussions with Anat Ninio and And Rosta. Neither of them agrees with my conclusions, but the conclusions would not have emerged without these discussions. Steve Abney also helped with information about parsers.

[1]Some of the most interesting types of apparent discontinuity are the complicated patterns discussed by Rosta (this volume), for which he suggests an analysis in terms of the new grammatical relation 'proxy'. His analysis seems much better than the one I offered in 1990 in terms of 'projections'.

same way as if they formed a continuous phrase, although other rules split the phrase into two parts. For example, the link between *raining* and *it* is just the same as in *It rained,* and illustrates the fact that the lexeme RAIN requires its subject to be IT; but the phrase consisting of the verb and its subject is split by the 'main verb', *keeps.*

Discontinuous phrases are a challenge for any linguistic theory, not because examples like (1)b are ungrammatical (that's easy to deal with, as we shall see), but because some examples are grammatical. Why are some good, while others are bad? Is there some general criterion, or does each pattern have to be dealt with piecemeal?

Before we start to answer these questions, let me introduce the two main 'families' of syntactic theory that are currently available. I shall call them 'phrase-based grammar' and 'dependency-based grammar', abbreviated to PG and DG. PG includes any theory that takes phrase-structure, alias constituent structure, as basic; so it dates back to Bloomfield 1933, but becomes especially clearly identifiable in Chomsky's Phrase-Structure Grammar (1957), and then in all the subsequent developments based on PSG - all Chomsky's own theories, Relational Grammar (Blake 1990), Lexical Functional Grammar (Bresnan 1982), Generalised Phrase-Structure Grammar (Gazdar et al 1986) and Head-driven Phrase-Structure Grammar (Pollard and Sag 1987). The PG family also includes other theories which grew directly out of the Bloomfield's work, of which the most influential nowadays is probably Systemic Grammar (Halliday 1985, Butler 1985). Categorial Grammar (Wood 1993) is on the boundary between PG and DG; since structures fit words together to make phrases it clearly qualifies as PG, but some linguists (e.g. Chomsky, pc) consider it to be an example of DG.

All these theories rest on the assumption that the basic relation in sentence structure is the part-whole relation between a word and whatever phrase it belongs to, so phrases are just as basic as single words. The touch-stone of this kind of theory is the Noun Phrase, which is entirely missing from traditional grammar and discovered (or invented) first by Bloomfield. Any grammar which defines the subject of a verb as a noun phrase belongs to the PG tradition.

The other family is a more direct extension of traditional grammar, in which the relations between individual words are the basis for sentence structure. In this kind of analysis, which many of us did at school, sentence analysis consists of finding the main verb, the subject, the object, and so on, each of which is a single word. The analysis also recognises modifiers for these words, and a more elaborate analysis allowed subordinate clauses to be accommodated as well, and the term 'phrase' was indeed applied to some combinations of words, but only to those which were rooted in a preposition or a non-finite verb. Neither the term 'phrase', nor the concept of a phrase, was at all central to this kind of analysis. This tradition is continued in modern DG, which is especially lively in central and eastern Europe. Well-known exponents include Mel'čuk (1979, 1988), who worked in Moscow before moving to Canada,

Sgall and his colleagues (1986) in Prague, and Kunze (1975) in Berlin. The classic text was written by the Frenchman Lucien Tesnière (1959), who was also a member of the Prague Linguistic Circle. My own theory, Word Grammar (Hudson 1984, 1990, Fraser and Hudson 1992) is part of the DG tradition, though my first work as a linguist was firmly embedded in the PG tradition - in Systemic Grammar, in fact.

One of the main differences between these traditions lies in their treatment of discontinuous constituents. Most versions of PG are based on Chomsky's formalisation of PSG[2], according to which discontinuous constituents are impossible in principle. This principle follows if phrase-structure is equivalent to a bracketing of the string of words, and of course a bracket which encloses the whole phrase must also enclose any words which make it discontinuous. If we bracket our first example sentence like this: *(It keeps raining),* we correctly show that *it* and *raining* belong to the same phrase, but incorrectly we also include *keeps.* It is impossible to use brackets to show discontinuous phrases. This forces two further claims on the theory: first, that discontinuous phrases are normally ungrammatical, and second, that where an apparently discontinuous phrase is grammatical, it must be reanalysed in such a way that the discontinuity disappears (e.g. by recognising two distinct, but continuous, analyses related by transformation). I shall discuss these claims below, arguing that the first is correct but the second is false.

The DG approach has never been given a clear mathematical formalisation comparable with Chomsky's work on PSG[3], and in particular it has never been accepted that a DG analysis should be expressible in terms of brackets. The status of discontinuity has been a matter of debate and difference, as dependency relations are more abstract than word-order relations. Some DG linguists keep their dependency analyses at this level of abstraction and don't concern themselves closely with the ways in which their structures are mapped onto more concrete observables such as word order. This means that it is possible to develop a DG theory in which discontinuity is normal and allowed. Some DG practicioners (e.g. Kunze 1982, Covington 1990) have been impressed by the existence of grammatical but discontinuous phrases, such as the ones discussed above, and have concluded that DG

---

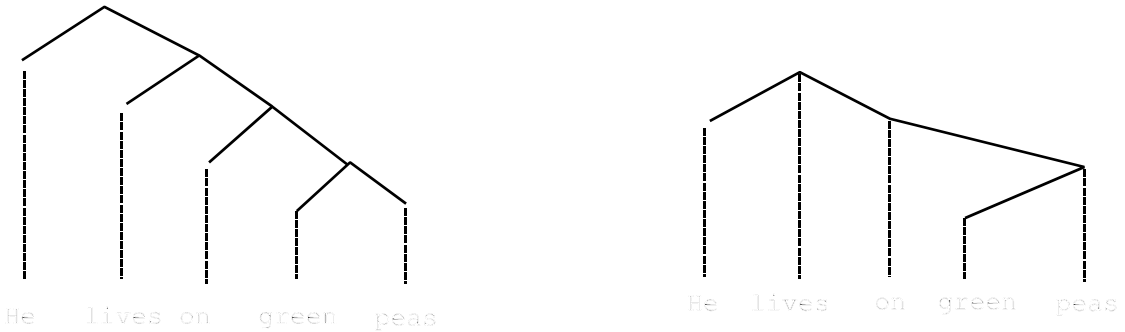
[2]McCawley has often pointed out (e.g. 1988:39f) that if Chomsky had formalised PSG in terms of trees, rather than brackets, discontinuous phrases would have been permitted because the relevant branch of mathematics is graph theory, which allows much richer structures.

[3]It's true that Gaifman (1965) showed that it was possible to formalise DG in such a way that it was equivalent to PSG, but his formalisation seems to have lost some of the basic insights of the DG tradition, such as the essential independence of a word's various dependents, and the optionality of word-order restrictions. Later work in DG has certainly not been based on Gaifman's formalisation in the same way that PG work has rested on Chomsky's formalisation of PSG.

should allow discontinuity in all cases. This approach seems to raise far more problems than it solves, because it leaves us without an explanation for the badness of examples like **He lives green on peas,* and an infinity of similar cases; my own view is that PG is right in assuming that phrases normally do have to be continuous. However it is important to bear in mind that the basic assumptions of DG don't force this assumption.

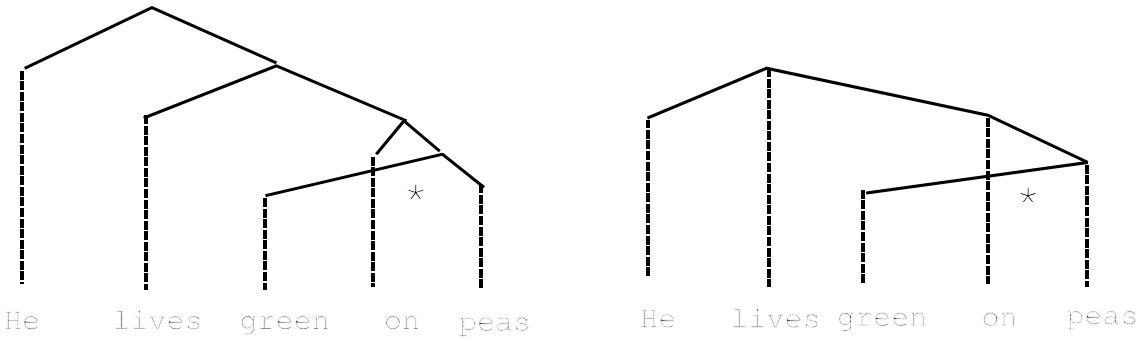
Most DG theorists probably accept what is called the principle of 'projectivity', which rules out all discontinuous phrases just as firmly as in PG. To explain this principle and its name, I need first to explain the standard notation for DG, Tesnière's 'stemma'. The stemma is a tree like a Chomskian phrase-marker, but of course its nodes are all words, rather than phrases; and as in many phrase-markers, the words are 'projected' onto nodes in the tree by vertical lines (which in this diagram are dotted). Here are a phrase-marker and a stemma for *He lives on green peas.* I have deliberately minimized the differences by ignoring unary branching in the phrase-marker.

(3)



(In the stemma, the vertical dimension shows subordination, so heads are connected upwards to their heads.) This stemma is 'projective' because it is possible to project each word up to its node without crossing any dependency lines in the stemma. In contrast, it is impossible to draw for a discontinuous phrase either a phrase-marker or a stemma in which no lines tangle. This can be seen in the next diagram, for **He lives green on peas.*

(4)



The assumption of projectivity makes DG exactly equivalent to PG as far as discontinuous phrases are concerned. This is a matter of surprise for some PG linguists, whose respect for PG rests in part on the assumption that PG is the only approach which explains why phrases 'hang together'. For example, why is linguistic competence 'structure dependent', so that small children realise that rules for positioning the subject of a sentence apply to the whole subject phrase, and not just to the root noun? The DG and PG answers are the same: because the alternative is tangled structure lines. For example, the interrogative equivalent of a) below has to be b), because c) contains a discontinuous phrase, *long linguistics ... books,* and tangling is inevitable.

(5) a Long linguistics books are expensive.
b Are long linguistics books expensive?
c *Long linguistics are books expensive?

The uncontroversial claim that words are grouped into phrases, which tend strongly to be continuous, is not evidence for PG as opposed to DG, but for either PG or DG with projectivity - or an equivalent - as opposed to DG without projectivity or any equivalent restriction.

The crucial difference between DG and PG, then, is that projectivity is fundamental to PG, and arises from its initial premises, whereas it is grafted onto DG by some (but not all) practicioners. This means that it is subject to debate and modification in DG, but not in PG. Granted that something like projectivity is needed, in order to rule out the countless cases of ungrammatical discontinuity, what precisely do we mean by 'something like projectivity'? And given that some discontinuous phrases are in fact grammatical, is it possible to formulate an alternative to classical projectivity which strikes just the right balance between strictness and flexibility - strict enough to exclude all the bad cases, and flexible enough to allow all the good ones?

My aim in this paper is to propose a principle which has just this ideal combination of virtues, called the No-tangling Principle. I shall have to prepare for it by discussing some elementary characteristics of dependency structures, but once I have introduced and explained it, I shall apply it in two different ways. First, I shall show how it explains some otherwise puzzling facts about adjuncts and extraction in English, and second, I shall outline its implications for the design of a parsing system. This will show how efficiently the No-tangling Principle organises the words in a sentence. Finally I shall show how the No-tangling Principle explains the discontinuities in the various English constructions that I listed above - subject-raising, extraction, tough-movement and extraposition.

One important gap in the discussion is where coordination ought to be. Coordination does lead to discontinuous phrases (e.g. in *He speaks and writes French,* the phrases rooted in the verbs are *He speaks .. French* and *He .. writes French),* but the special issues raised by coordination are tangential to the ones I am discussing here, and seem to remain much as I have suggested elsewhere (Hudson 1984:211ff, 1988a, 1989a, 1990:404ff).

2 The geometry of dependency structures

Most DG grammarians define sentence structure as a projective stemma of the kind illustrated above. The classic statement on DG stemma geometry is in Robinson (1970), who allows only one head per word, only one word (the sentence-root) without a head, and no loops. Only phrase-marker-like structures are allowed by this definition, when combined with projectivity (which Robinson calls 'adjacency', a term which I adopted in earlier work but now regret). It is hard to see how to reconcile it with the existence of well-formed discontinuities, and for this reason (among others), some DG linguists allow multi-level analyses which include an abstract level at which discontinuous phrases are shown as continuous.

My own reaction has been to develop a richer DG theory, Word Grammar, which stands to classical DG in much the same way that Gazdar's Generalised Phrase-Structure Grammar stood to classical PSG. Gazdar's aim was to accommodate non-standard structures such as our discontinuous phrases, and his solution was to enrich PSG in various ways, notably by greatly expanding the kinds of feature-categories that could be used. My aim is similar, but my solution is to enrich the theory by allowing extra dependency relations. Every dependency has to be justified by one or more rules in the underlying grammar, but there is no limit in principle to the number of heads a single word may have; nor is there a general ban on loops. In short, there are no general limits to the number or kind of dependency relations (which means, of course, that the traditional stemma can no longer carry all the dependencies, so a new

notation is needed). In the course of this paper we shall see numerous examples of words which have several heads, and also of dependency chains which form a loop (A depends on B .... which depends on A). The primary reason for allowing such structures has been factual: there was no way of defining the structures concerned without recognising them. But now they are in place, we can exploit them in the search for a more suitable replacement for the Projectivity Principle.

I shall illustrate the discussion with examples of subject-to-subject raising, as in *It is raining.* We have to justify three dependency relations:

it depends on *is.*
raining depends on *is.*
it depends on *raining.*

Each of these taken individually is more or less uncontroversial.

■ *It* is the subject of *is,* which is one kind of dependency relation. (Virtually all DG analyses take the verb as the root of the sentence, with the subject and object as its dependents.) A host of rules connect *it* to *is* - rules for so-called 'agreement', for inversion, and so on. This dependency, at least, is beyond question.

■ *Raining* is the complement of *is.* Virtually all modern linguists recognise the tensed verb (*is*) as the root of the sentence, with the next (non-finite) verb as its complement. This helps to explain for example why *is* can occur without *raining (It is!),* but not vice versa, and why *raining* is a participle, rather than, say, an infinitive (as it would have been as complement of *will).*

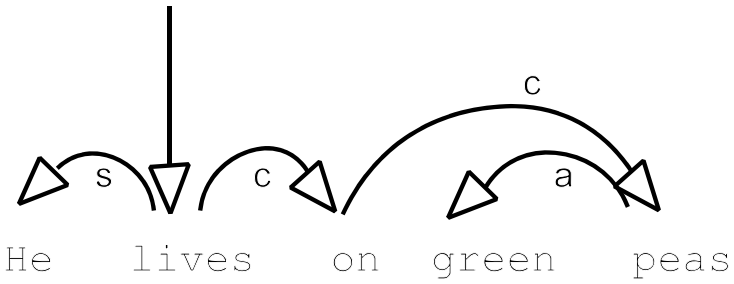
■ *It* is the subject of *raining.* This is why it cannot be replaced by any other word, such as *something,* so **Something is raining* is bad for just the same reason as **Something rained.* All the familiar evidence for subject-raising supports this analysis.

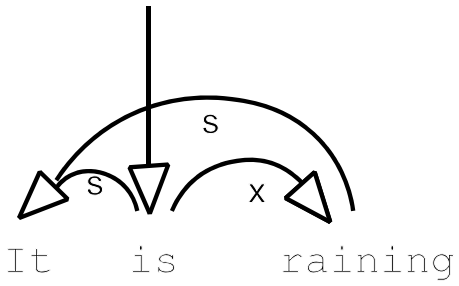
But when we put the three dependencies together, we can no longer fit them into a standard projective stemma because the dependency from *it* to *raining* inevitably tangles with the projection from *is* to its node; and in any case *it* has two heads.

However convenient the standard 'stemma' notation may be for simple dependency structures, it is too rigid for more complex ones. The real crunch arises when we need to recognise mutual dependency and loops (as we shall below), so we can anticipate the problems they will raise by changing to standard Word Grammar notation now. The main change is to use arrows instead of the vertical dimension to show the direction of dependency. The arrow points towards the dependent; and since

different kinds of dependent (e.g. subject and object) are distinguished by labels on the arrows, it may be helpful to think of the arrows as signposts pointing in the direction of the named dependent. Here are WG-style dependency structures for two sentences, with a key to the function labels. (The term 'xcomplement' is borrowed from LFG (Bresnan 1982); in my own earlier work I have tended to use 'incomplement', meaning 'incomplete complement', with the same meaning.)

(6)

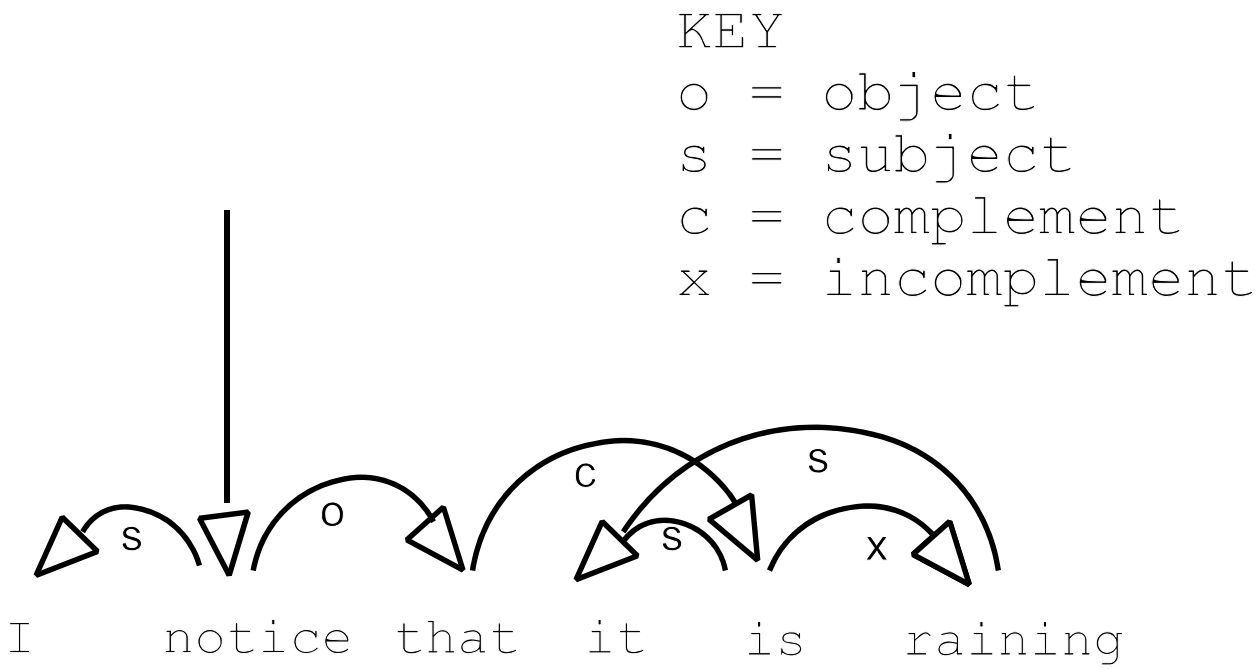




KEY
s = subject
c = complement
a = adjunct
x = xcomplement

The vertical arrow pointing straight down at the sentence-root shows that this has no head inside the sentence, but that in principle it could have had a head. For instance, we can make *is* into a dependent of *that,* as in *I notice that it is raining:*

(7)



This 'virtual head' is important in a discussion of discontinuity, because it makes discontinuity show up as potential tangling even when there is no actual tangling.

These diagrams illustrate the theoretical context for the discussion in the rest of the paper. Every word has at least one head (which may be 'virtual'), but may have more than one, and some dependency arrows are allowed to tangle with each other. The question is how to distinguish permissible tangling from the tangling that leads to ungrammaticality.

3 The No-tangling Principle

What is it about the structure of *It is raining* that makes tangling acceptable, in contrast with its badness in **He lives green on beans?* The crucial difference is that the arrow to *green* causes tangling but is essential for 'connectedness', i.e. for linking *green* to the rest of the sentence. If we ignore this arrow, the sentence falls to pieces because the structure is what we shall call 'incomplete', meaning that there is a word with no head, and (therefore) no linkage to the root-word. In contrast, the tangling arrow which links *it* to *raining* can be ignored because *it* has another head, *is.* If we ignore the tangling arrow the structure is still complete. This example is typical and we can make a very simple generalisation about (geometrically) well-formed

dependency structures: they all contain a sub-structure of dependencies which is free of tangles and which connects all the words together.

This insight has been developed independently in two different ways. Rosta (this volume) defines 'dependency' as a derivative relationship, superimposed on the more basic grammatical relations, but defines it in such a way that only the dependencies in this tangle-free sub-structure are recognised at all. According to this view, all dependencies are (by definition) tangle-free. The other approach treats all grammatical relations as subtypes of dependent, as in the published versions of WG, but allows some dependencies not to 'count' as far as the anti-tangling ban is concerned. This is the approach I shall follow here. At present it is hard to see whether these theories are substantively different, but the difference between them is certainly worth exploring. In any case, they both replace a series of different explanations that I have tried over the years for the difference between good and bad discontinuities (Hudson 1984:98, 1990:117). Rosta (this volume) points out a fundamental flaw in my 1990 attempt.

The basis for my account is the No-tangling Principle:

(8) The No-tangling Principle
The dependency structure for a phrase must contain a substructure which is tangle-free and complete[4].

We can call the tangle-free and complete substructure the 'structural skeleton'; *It is raining* has a structural skeleton, but **He lives green on beans* does not. In structure diagrams we can pick out the skeleton by highlighting the arrows that comprise it. The word *phrase* is used in its general sense, of course, so it includes sentences as well as smaller phrases.

One of the noteworthy characteristics of the No-tangling Principle is that it applies globally to the whole structure and not just to local areas. This has the interesting consequence that under certain rather special circumstances, a local configuration which is grammatical in one larger context may be ungrammatical in a different one. An example of this is provided by the data in (9).

(9) a What will happen?
b Tomorrow what will happen?
c I wonder what will happen.
d *I wonder tomorrow what will happen.
e Tomorrow I wonder what will happen.

[4]It is tempting to separate out a separate principle of completeness, called the 'No-dangling Principle'.

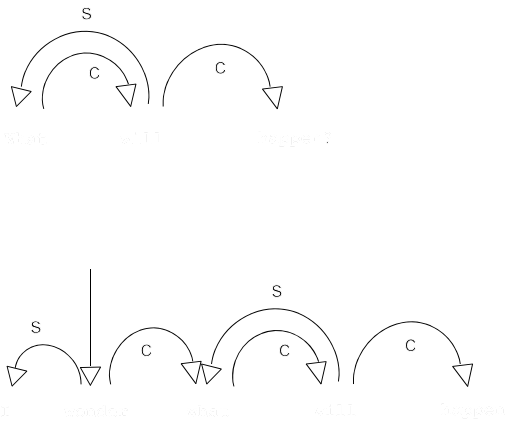
Why can't the clause in b) be subordinated to *I wonder,* in just the same way as the one in a)? The answer may be intuitively obvious, but it is not at all obvious how a generative grammar (such as WG) should exclude d) while allowing all the others[5].

The explanation involves two elements: the No-tangling Principle, and the WG analysis of wh-interrogatives. Wh-interrogatives illustrate the need for loops in the dependency structure, and indeed this is the crux of our explanation, so we must start by demonstrating that, on the one hand, *what* depends on *will,* but on the other hand *will* depends on *what.* The first of these dependencies is obvious, because *what* is clearly the subject of *will,* and as explained above, subjects are a kind of dependent. The second dependency is less obvious, but easily justified from examples like (9)c - *wonder* requires an interrogative word as its complement, so its complement must be *what.* Furthermore, it is *what* that requires *will* to be a tensed verb (though *to* + infinitive would also have been possible), which is a clear indication that *will* is its complement. This is confirmed by the fact that *will* (and the rest of the subordinate clause) can be omitted by 'sluicing', as in *I know something exciting is going to happen, but (I wonder) what?* It is hard to avoid the conclusion, then, that *what* and *will* are linked by two dependencies, each going in the opposite direction: one with *what* as the subject of *will,* and another with *will* as the complement of *what.*

[5] A tempting Chomskyan explanation runs as follows: *what will happen* is the complement of *wonder*. When *tomorrow* is extracted it is attached as an adjunct of *what will happen,* but adjuncts are not allowed to attach to arguments, i.e. to complements or subjects of verbs (Chomsky 1986:6); therefore *tomorrow* can't attach to *what will happen.* One weakness with this explanation is that *tomorrow* can move up to the front of the whole sentence: *Tomorrow I wonder what will happen;* if this displacement is due to the normal processes of extraction, how can *tomorrow* 'hop' across the boundary of its own clause without adjoining to it or in some other way being attached to it so as to leave a trace? Another weakness is that the same restriction on extracting *tomorrow* applies to any tensed subordinate clause, regardless of whether it is an argument or not:

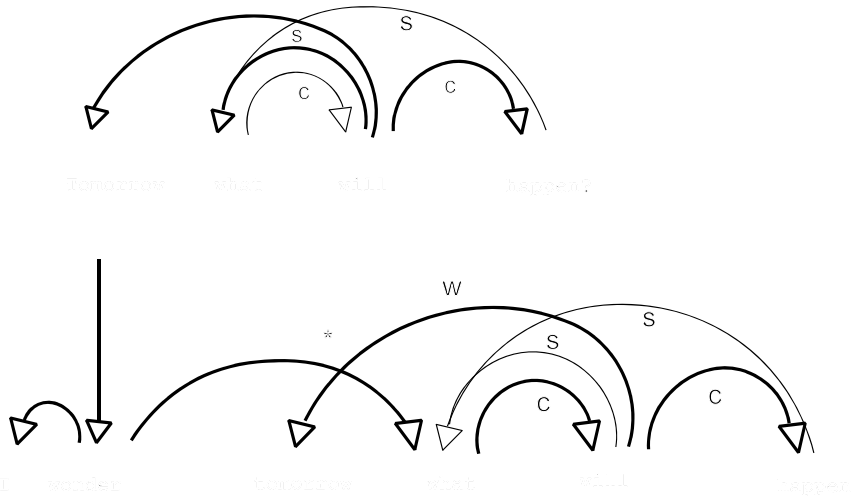
(i) I can't answer the question (*tomorrow) who will come.
(ii) I'm working now (*tomorrow) because I have to teach.

(10)



Suppose we now want to attach *tomorrow* as a dependent of *will.* If the only head that *what* has is *will*, there is no problem: the line from *tomorrow* to *will* simply goes round the one from *what* to *will.*

(11)



But if *what* also depends on *wonder,* tangling is unavoidable. Without *wonder*, we can ignore the fact that *what* has a 'virtual' head, as the only potential link to outside words; this is possible because *what* already has *will* as its head, so the need for a head is satisfied. The addition of *wonder* adds a dependency that really does tangle with the one from *tomorrow* to *will,* and neither of these dependencies can be ignored without losing completeness.

This example gives interesting support to the basic idea behind the No-tangling Principle, which is that only a subset of the dependencies in a sentence are relevant

to tangling, and this subset can only be recognised when the total structure is taken into account. This conclusion might suggest that the ban on tangling would be of little help to the processor concerned with the current word rather than with the total sentence; after all, such a restriction doesn't help me if I have to wait till the end of the sentence before applying it. However we shall see that this conclusion doesn't follow in practice because the constructions that give rise to tangling are such that the first dependency established is always tangle-free in an absolute sense.

4 An incremental parser for Word Grammar

One of the attractions of the No-tangling Principle is that it is easy to integrate into a model of parsing, a fact which may help to explain why it exists - i.e. why our language ability has developed in such a way that we all 'know' and apply it. The general point is that the No-tangling Principle helps the listener to find the syntactic structure of a sentence by restricting the range of structures that are worth considering. The same can be said, of course, about any general theory of syntactic structure, but the benefit of the No-tangling Principle is that it forces us to build a tangle-free skeleton which is equivalent to a phrase-marker (with only continuous phrases), while at the same time allowing us to add, at very little extra cost, whatever extra dependencies are needed for discontinuous structures.

The aim of this section is to outline one parsing algorithm which can be used efficiently to recognise dependency structure without tangles. It was first described in Hudson (1989b). Fraser (1993) surveys a large number of other algorithms for dependency parsing which have been developed and applied by other workers. The one described here has advantages over some of these, but may not be the best possible, and is certainly not complete - as can be seen from the lack of discussion of ambiguity and coordination (though in my defence my 1989 parser allows all possible readings, and does deal with coordination).

Before introducing the algorithm I shall point out some limitations on how a human parser, such as me, might process the words in the following example, assuming that the words were presented and processed 'incrememental ly', i.e. one at a time.

(12) Really bad term papers by students about syntactic theory depress me.

Suppose I have just heard the words *really bad*. I work out that *really* depends on *bad,* and I record this information. But this decision will affect the way I analyse later words, because *really* is no longer available for any kind of dependency links. Why?

Because any dependency arrow linking *really* to a later word is bound to tangle either with the one that will eventually link *bad* to the sentences's root, or with the root-word's vertical 'virtual' arrow.

(13)

As can be seen from this diagram, all three positions A, B and C will be blocked off by some dependency. This is because of the abstract geometry plus the No-tangling Principle, and not because of some special fact about *really,* so we can generalise to all words that depend on a following word: any such word is inaccessible to any future dependency. We can show this in our running record of the parsing process by putting inaccessible words between parentheses:

<(Really) bad>.

My parsing effort continues. I recognise *term,* try to link it to earlier words, and fail. The result is that I simply add it to the running record, giving:

<(Really) bad term>.

Then comes *papers,* which I can link to both *term* and *bad* (but not, of course, to *really).* However, it's essential for me to link *papers* to *term* before linking it to *bad,* because the analysis will be wrong if I establish the latter without the former. Suppose I were to construct the partial analysis shown below:

(14)

Sooner or later *term* has to have a head, so if its head hasn't appeared before *papers,* it must follow (like X in the diagram); but in that case the dependency link will inevitably tangle with the one between *bad* and *papers.* Once again this is a 'geometrical' fact, which can be generalised: I mustn't link the current word to any earlier word until all the intervening words have heads. Applied to the current example, then, this means that when I hear *papers* I first take *news* as its dependent, and then I do the same for *bad.* Since these are both dependents followed by their heads, they disappear from sight, as it were:

<(Really bad term) papers>

In all the links established so far the earlier word has depended on the later word. We can call the process of recognising these links 'capture' - the current word 'captures' an earlier word as its dependent. We now look at an example of the converse of this arrangement. I hear *by,* and try to link it to the only accessible preceding word, *papers.* I succeed, but this time I recognise the earlier word as the head of the current one. Let's call this operation 'submission', in recognition of the subordinate status that the current word adopts in relation to the earlier one. This process does not affect the status of the head word (*papers),* which is still available for receiving later dependents and its own head, so here is the new running record:

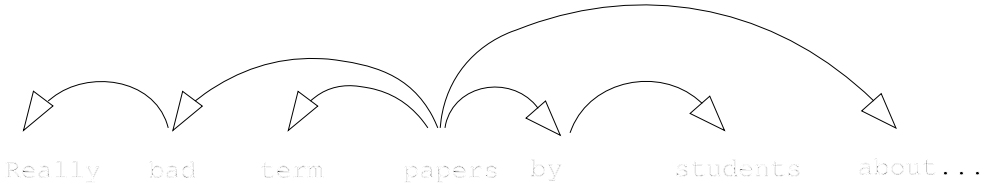
<(Really bad term) papers by>

Another application of submission integrates the next word too, once again without affecting the status of earlier words:

<(Really bad term) papers by students>

But as soon as I do the same to *about,* we notice that I do affect the status of the preceding words. Here is the structure so far:

(15)



What happens when I take *papers* as the head of *about* is that I isolate *by students* from the rest of the sentence, because any dependency link between either of these words and a later word must, inevitably, tangle with the one between *papers* and *about.* More generally, a dependency between the current word and its head hides all the words between these two words, except for the words in the dependency chain that links them. The running record is therefore as follows:

<(Really bad term) papers (by students) about>

The rest of the sentence is straightforward, and can be shown as a sequence of captures, submissions or simple additions:

syntactic just added:
<(Really bad term) papers (by students) about syntactic>
theory captures *syntactic:*
<(Really bad term) papers (by students) about (syntactic) theory>
depress captures *papers:*
<(Really bad term papers by students about syntactic theory) depress>
me submits to *depress:*
<(Really bad term papers by students about syntactic theory) depress me>

The effect of the No-tangling Principle has been to reduce very considerably the number of words that the parser needed to consider at each stage as possible candidates for dependency links. For example, if another word had been added at the end of the sentence, only two words are candidates for its head: *depress* and *me.*

It is easy to express these processes in a 'shift-reduce parser', a standard type of parser[6] whose purpose is to take the next word and somehow combine it with the stack of words already processed. 'Shifting' is the process of adding the next word to the top of the stack, and 'reducing' covers whatever operations are available for grouping words together. We start with shifting, in which the next word is removed from the list of unprocessed words and added to the stack:

(16) SHIFT {<Stack>,[NextWord+Rest]} = {<Stack+[NextWord]>, [Rest]}

[6]A very accessible and brief explanation of shift-reduce parsers can be found in Pulman 1992. Fraser (1993) describes a wide variety of shift-reduce parsers for dependency grammars.

Then we start reducing, by first trying to capture one or more dependents, and then trying to submit to a head. Either of these operations may be unsuccessful, leaving the stack unchanged, but it is not possible to capture after submitting.

(17) REDUCE X = Y if
1. CAPTURE X (recursive) = X', or X = X',
2. SUBMIT X' = Y, or X' = Y.

We now move to the definitions of CAPTURE and SUBMIT. In these formulae all the variables (named by letters) may be zero, with the obvious exception of the current word Z and B, the earlier word which is to be linked to Z.

(18) CAPTURE <..[(A..)B..C], [(Y..)Z]> = <..[(A..B..C Y..)Z]>
and B depends on Z.

This formula uses parentheses as I did above, to fence off words which are inaccessible for later dependencies; as explained, the effect of taking B as a dependent of Z is to make B and all intervening words inaccessible. The formula for submission is similar, though not identical.

(19) SUBMIT <..[A..B..C], [(Y..)Z]> = <..[A..B(..C Y..)Z]>
and B is the head of Z.

In this case the effect of the link is to parenthesise all the words between B and Z, though not B itself (or any words before it).

These formulae correspond to the processes I described above, but they assume word-groups which I didn't mention before, namely the strings of words which have so far been linked by dependency. These are shown by the square brackets and could be called 'pseudo-phrases' because although the left bracket corresponds to a genuine phrase boundary, the right bracket need not. Moreover some phrases (e.g. *by students*) are never bracketed, and at the end of the process all internal brackets will have been removed. In short the role of pseudo-phrases in this parser is only very indirectly related to that of real phrases in sentence-structure, and should not be seen as evidence for PG! I shall return to this point below.

The following lists show how the stack would have grown under the control of this parser:

(20)
<[Really]>
<[(Really) bad]>
<[(Really) bad] [term]>
<[(Really bad term) papers]>
<[(Really bad term) papers by]>
<[(Really bad term) papers by students]>
<[(Really bad term) papers (by students) about]>
<[(Really bad term) papers (by students) about] [syntactic]>
<[(Really bad term papers by students about syntactic theory)]>
<[(Really bad term papers by students about syntactic theory) depress]>
<[(Really bad term papers by students about syntactic theory) depress me]>

The main point of this discussion was to show how the No-tangling Principle can be accommodated in a parsing algorithm and how it improves the efficiency of the system by reducing the number of words that need to be considered when looking for the head and dependents of the current word. I am not, of course, claiming that this is a virtue unique to DG; far from it, as the restrictions noted above are very similar to those which come from a standard shift-reduce parser based on PG. However there are some differences between the two approaches.

■ As mentioned above, the brackets are not part of the sentence structure itself, but are simply a record of the parsing process. The sentence structure consists of nothing but the word-word dependencies which are established each time SUBMIT or CAPTURE is applied. These dependencies are preserved, but all the brackets disappear by the end of the parsing; and in any case the connection between the brackets and genuine phrases is only very loose.

■ The parser does not recognise phrases as such, but only individual dependencies, so it reduces any pair of words which can be linked by a dependency even if the same head could have further dependencies. If further potential dependents are found later, they can simply be clipped onto the existing head without having to 'reopen' a phrase which had been completed. For example, in parsing *papers by students about syntax*, the process of adding *by students* to *papers* does not form a complete phrase, which then has to be either reopened or duplicated (for adjunction) in order to accommodate *about syntax*. Such cases are a well-known problem for PG parsers, which have led to the development of a new class of parsers called shift-attach parsers (Abney 1989).

■ Even more problematic for a PG parser are cases where a head word takes two complements (e.g. GIVE). If the requirement for shifting is that the whole phrase's

structure be recognised, it follows that the first complement cannot be recognised until the second complement is found; only then can the requirements of the head be satisfied. A WG parser avoids this problem by recognising each dependency as soon as a word is found that satisfies whatever criteria apply to that particular dependency.

In all these respects the WG parser seems to predict human behaviour better than at least some PG-oriented parsers.

5 Parsing of subject-raised phrases

How does this parser cope with the only kinds of discontinuous phrases that we have considered so far, those due to subject-raising? The answer is surprisingly simple.

The description of the parser given above ignores all the activity that is involved in deciding whether or not a pair of words can be linked by dependency. For example, I took it for granted that *really* could be taken as the dependent of *bad*, but this has to be the conclusion of a certain amount of work by the parser, in which the properties of these two words are checked in the grammar to make sure that they are compatible in this relationship.

Another aspect of this activity is that, in some cases, information flows from one word to the other once the link is established. The obvious example of this is agreement between words that are only partially specified; for example, between *these* and *sheep*, or between *the* and *cows*. In both cases the grammar requires the determiner to have the same number as the common noun, and in the first example the observable plurality of *these* transfers to *sheep*, while in the second example the information flows in the opposite direction. As soon as the parser links the common noun to the determiner, whatever is known for sure about the number of one automatically applies to the other as well, by simple logical deduction: the number of *these* is plural, the number of *these* must be the same as that of its common noun, namely *sheep*, therefore the number of *sheep* is also plural.

Now consider the parsing of *It is raining*. Suppose the parser has finished *It is*. One thing it has discovered (from the grammar) about *is* is that it shares its subject with its xcomplement. When it processes *raining* it recognises this word as the xcomplement of *is*, so it then follows through all the logical consequences of this conclusion. One of these is the inference that since *it* is the subject of *is*, and *is* shares its subject with its xcomplement, namely *raining, it* must also be the subject of *raining*. The similarity to the logic behind number agreement is striking, but in this case what is added is an extra dependency, between *it* and *raining.*

The conclusion, at least as far as subject-to-subject raising is concerned, is that the discontinuous, tangling, subject link is added by simple logical inferencing

applied to information that is easily available in the grammar. All this work can be done without looking at the rest of the sentence, and follows automatically as soon as the link between the verb and its xcomplement is established. We shall see that something similar is true of all discontinuous structures, though in other cases the parser has to do a little more work to choose among alternatives offered by the grammar.

Another question about parsing involves the No-tangling Principle itself. If it applies globally to the total sentence structure, as I argued in the last section, how can it control the operation of a parser which is building the structure incrementally, one word at a time? Does the parser need to stand back and review the total structure at the end of the sentence? It's true that a final check on completeness is necessary, in case some word has been left without a head. This shows up in the parser if the stack holds more than one list. A successful parse reduces all the words on the stack to a single list. But as far as tangling is concerned, there is no need for a final check because the design of the parser guarantees that all the dependencies due to SUBMIT and CAPTURE are free of tangles. In essence, then, these two operations build the structural skeleton, and the logical inferencing mentioned above adds the remaining dependencies, some of which cause tangling.

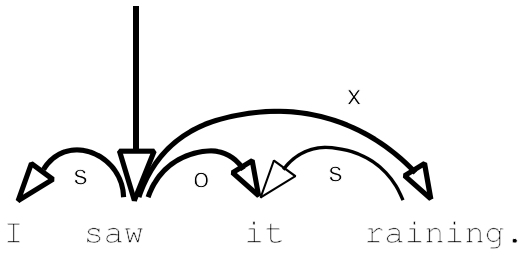
If the parser guarantees satisfaction of the No-tangling Principle, and this principle explains the workings of the parser, one wonders to what extent they are separate. If the grammar is applied via the parser, the No-tangling Principle could be dropped without affecting the grammar's generative capacity; and if the No-tangling Principle is preserved, it presumably limits the range of possible parsers, and maybe the parser defined here is the only candidate. However, I see no advantage in jettisoning either the principle or the parser at this stage, even if they are different descriptions of the same facts; if they complement and explain each other, so much the better.

Before moving on to other discontinuous structures, I should explain that subject-raising applies in just the same way even if the order of elements is different. The order of the first subject link and the shared head is reversed in two constructions: where the subject and auxiliary are inverted, and in subject-to-object raising. The first dependency to be found is still the one that belongs to the skeleton, but in these cases the second one does not in fact lead to tangling. In the diagrams below for *It is raining* and *Is it raining*?, it can be seen that the dependencies are exactly the same, in spite of the different orders.

(21)

The other kind of raising is in examples like *I saw it raining* or *I expect it to rain,* where I shall simply assume without argument that *it* is the object of the raising verb (though I recognise that this is controversial). The analysis shows the similarities to subject-to-subject raising by applying the same function, xcomplement, in both cases.

(22)



6 Extraction

Extraction is handled in much the same way as subject-raising. In both cases, the parser first establishes a semantically neutral 'holding' dependency to link the displaced word to the skeleton, and then adds the semantically important dependencies later by more or less automatic rules which are triggered by the first dependency. For subject-raising the holding dependency is either subject or object, but in extraction we can call it 'extractee'[7]. To avoid confusion with the 'x' of 'xcomplement', and to remind ourselves that typical extraction involves wh-words, we label it 'w' in diagrams.

How is the extractee dependency recognised by a parser? There are two separate cases, according to whether or not the extractee itself gives away the fact that it is an extractee - wh-words do, but ordinary topicalised objects and so on don't. Take a wh-word like *what,* which takes a tensed verb as its complement. In the examples given so far it has been the subject of this complement verb (e.g. *I wonder what will happen*), but another possibility is for it to be the verb's extractee (e.g. *I wonder what*

[7]In earlier work I have tended to call the extractee 'visitor', but 'extractee' is easier to remember.

he did). These two possibilities are laid out in the grammar, as facts about interrogative pronouns in general[8].

(23) a An interrogative pronoun has a tensed verb or TO as its optional complement.
b An interrogative pronoun may be either subject or extractee of its complement.

The other case has to be treated as a default operation performed by the grammar. Take an example like *Beans I like*. Nothing about *beans* tells us in advance that it should be an extractee. The parser does not discover this until it reaches the sentence root, *like,* and finds that *beans* still has no head. This is serious, because the sentence root is like a watershed in the parsing because no skeleton dependency can cross it. Every word before the sentence root must therefore have a head, so the parser uses 'extractee' as a kind of emergency dependency for linking otherwise unconnected words. This is reflected in the grammar simply by allowing any word to be the extractee of a finite verb (i.e. of a tensed verb or an imperative).

(24) A finite verb has an optional extractee, which may be any kind of word.

It may be that the parser should be prevented from using this default option until it has tried other possibilities, such as subject. This would explain the obligatory position of extractees before subjects, but it raises deep questions about the relation between the parser and the grammar which are better ignored here.

Whichever of these two routes leads to the original extractee dependency, the rest of the story is the same (as in all other accounts of extraction). Take *Beans I like.* Once the parser has sorted out the relations between *like* and the earlier words, it moves on - but finds there is nowhere to move to. This leaves it without an object for *like,* but it knows that *like* does have an extractee. It finds a general rule for extractees, which allows an extractee to have a second function from among the list of 'complement' functions (including 'object').

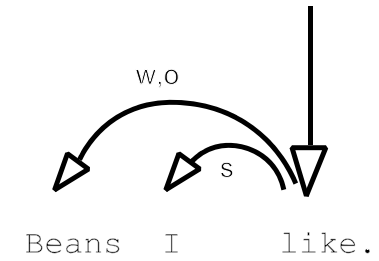
(25) A word's extractee may also be its complement.

Therefore it concludes that the extractee, *beans,* is the object of *like.*

---

[8]The complement of an interrogative pronoun is optional. This explains not only why 'sluicing' is possible (e.g. ...*I wonder what.*), but also why such pronouns can remain 'in situ', as in multiple questions (e.g. *Who ordered what?*) or echo questions (e.g. *He did what?!*).

(26)



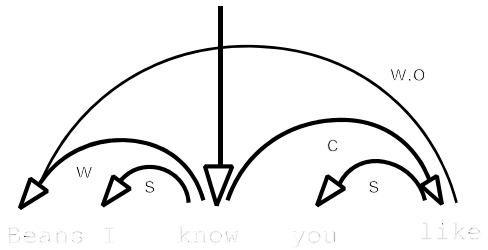
Exactly the same process allows it to take *what* as the object of *did* in *I wonder what he did.*

Neither of these examples involves discontinuity or tangling, but discontinuity does arise in slightly more complicated cases. Interestingly, in some of these cases there is no tangling, but since they are fully grammatical this is not a problem. Let's start, though, with a tangling case, *Beans I know you like.* To handle this example the parser has to know the second main rule for using extractees:

(27) A word's extractee may also be the extractee of its complement.

This rule becomes relevant once the parser has identified *like* as the complement of *know,* as it allows *know* to share *beans* with *like.* The effect of this rule is exactly analogous to that of subject-raising, which allows a raising verb to share its subject with its xcomplement.

(28)

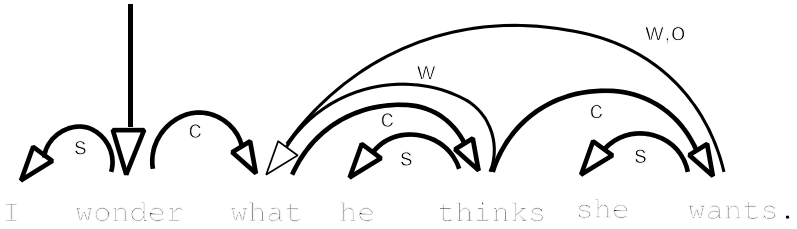


In this example there is a discontinuous phrase *beans ... like,* and as expected there is also tangling of dependencies, just as in the subject-raising examples.

For discontinuity without tangling we turn to the other kind of example, where extraction is triggered by some kind of 'link-word' such as a wh-pronoun. In such cases tangling is avoided because of the mutual dependency between the extractee and its complement verb (e.g. between *what* and *will*), which allows the extractee itself to act as the root of its phrase. Take an example like *I wonder what he thinks she wants.* As explained earlier, the parser has to take *what* as the extractee of *thinks,* but

by rule (27) it can also use it as extractee of *wants,* which in turn allows it to apply rule (25), giving *what* the status of object of *wants.*

(29)



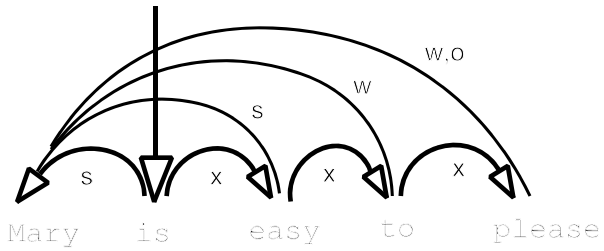
It should be clear how rule (27) applies recursively to allow 'long-distance' extraction, in just the same way that subject-raising allows a subject to be passed indefinitely far down a chain of raising verbs (e.g. *He has been happening to seem to keep working for a long time now.*). Extraction raises various other issues, such as island constraints and subject extraction, but these are irrelevant to discontinuity[9]. The main point I have tried to make here is that once the initial extractee dependency has been found, the rest of the analysis is very easy for the parser, though a few more options have to be considered than in the case of subject-raising.

7 Tough-movement

Tough-movement seems to share properties with both subject-raising and extraction. Like subject-raising it starts with a subject dependency; but like extraction, this dependency is copied into an extractee dependency (or something like it). For example, in *Mary is easy to please, Mary* starts as subject of *is,* which shares its subject with *easy* (by subject-raising); but when *easy* in turn shares *Mary* with its complement, *to,* the grammatical function changes to extractee.

[9]For detailed discussion of extraction in WG, see Hudson 1990:354ff, 1988b.

(30)



Tough-movement always produces discontinuity, as the moved object always moves across the triggering adjective to act as its subject, on its other side.

An open research question about tough-movement is whether it really does involve extraction. The similarities to extraction have often been noticed, but as Chomsky points out in detail (1982:308-314), there are also important differences. Rather tentatively I shall assume that the relation really is extraction, controlled by the same principles as in other constructions. I hope it may be possible to give some kind of pragmatic explanation for the relative difficulty of 'tough-moving' out of a tensed clause in examples like the following from Chomsky:

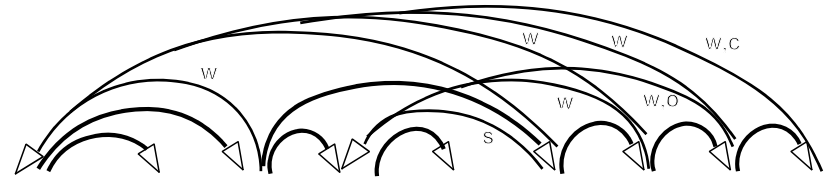
(31) ?This book is difficult to persuade people that they ought to read.

A discussion of discontinuity and tangling in dependency grammar would not be complete without at least some mention of the famous 'crossed dependency' examples like the next pair:

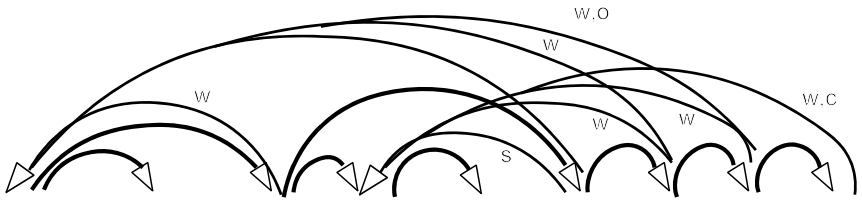
(32) a Which violins are the sonatas easy to play on?
b *Which sonatas are the violins easy to play on?

Unfortunately, the No-tangling Principle doesn't help in such cases, because whatever dependencies link the parts of the various discontinuous phrases are built on the same skeleton.

(33)



Which violins are the sonatas easy to play on?



*Which sonatas are the violins easy to play on?

It is at least tempting to assume that the explanation for these facts has something to do with the way in which we, as parsers, store information about extractees. Suppose we store them as a single stack, rather than as individual words. This immediately explains why (32)b is bad, because the first extractee to enter the stack *which (sonatas),* is also the first to leave it, contrary to the principles of stacks. In contrast the stack of extractees in (32)a grows as follows:

<>	before *are*
<which (violins)>	*are, easy*
<which (violins), the (sonatas)>	*to, play*
<which (violins)>	*on*

The same explanation also applies to other examples where two extractees are apparently carried forward together. Such examples can be quite awkward at best, but there is a clear difference between the following pair:

(34) a ?John is someone who I don't know what to make for.
b *Fudge is something that I don't know who to make for.

At the point of processing *to make,* the parser has an extractee stack containing <who, what> in the first case, and *what* is removed (as object of *make*) first; but in the second example the stack at this point is <that, who>, from which *that* i
s needed first.

Examples like these suggest strongly that a proper theory of parsing needs not only the No-tangling Principle, but also a stack for handling extractees[10]. It is paradoxical that our most general principle for avoiding discontinuity cannot explain the phenomenon which has been generally referred to, even in PG linguistics, as 'crossing dependencies'[11]!

8 Extraposition

The last discontinuous pattern that we shall consider is extraposition, and more specifically extraposition from NP as in the following:

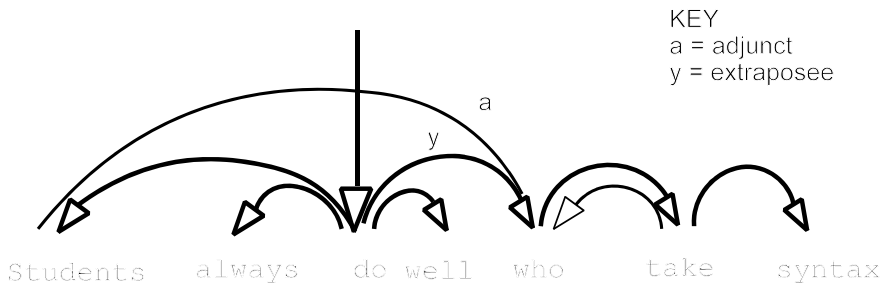
(35) Students always do well who take syntax.

This is different from the previous constructions because the discontinuity is caused by a 'movement' to the right, rather than to the left. However the same basic principles and explanations still apply. The skeleton contains a special semantically-neutral dependency called 'extraposee', which links the extraposed item to the previous verb, and guides the subsequent search for a semantically relevant dependency. We shall label the extraposee 'y', for no better reason than that this letter follows both 'w' and 'x' in the alphabet. As usual, the tangling is caused by a dependency which is not part of the skeleton:

[10]This general conclusion is due to And Rosta (pc).

[11]The other well-known crossed-dependency phenomenon is the pattern in Dutch and some dialects of German in which clause-final verbs occur in the same order as their subjects, rather than in the mirror-image order needed to avoid tangling. I offered a tentative explanation of these patterns in Hudson 1984:106-9, but I now think that the most plausible explanation will probably assume that all dependents of subordinate non-finite verbs can be raised to also act as dependents of their heads. However all the details remain to be worked out.

(36)



One question that arises in the analysis of this kind of extraposition is how best to define the range of extraposable items, and one of the advantages of DG is that it offers a choice between two kinds of definition: in terms of word-classes and in terms of grammatical functions. In this case the solution seems to be a functional definition, as extraposition can apply not only to relative clauses but also to content clauses (e.g. *The claim is often made that sentence-structure is based on phrases.*) and prepositions (e.g. *A book has just been published about the syntax of small clauses.*). What these examples have in common is their function, as adjuncts, or more precisely 'post-adjuncts', of the noun. This may explain why adjectives which in other respects are like relative clauses cannot be extraposed:

(37) a Students keen on linguistics often do well.
b *Students often do well keen on linguistics.
c Any student capable of answering this question is clever.
d *Any student is clever capable of answering this question.

Maybe these adjectives (*keen, capable)* are really 'pre-adjuncts', i.e. adjuncts which belong before their head, and are positioned after the noun only because they themselves have dependents (*on .., of ..)?*

The other major question concerns the nouns that can have their post-adjuncts extraposed. How can these nouns be identified structurally? This is an important question, because the more freely extraposition applies, the harder it will be for the parser to decide which noun to attach the extraposee to. One restriction is that the noun concerned must be a dependent, or near-dependent, of the verb to which the extraposee is linked. This is the 'Right-roof constraint', limiting extraposition to movement within the current clause, in order to explain the badness of examples like the third example below:

(38) a The fact that [students *who study syntax* always pass] is irrelevant.
b The fact that [students always pass *who study syntax]* is irrelevant.
c *The fact that [students always pass] is irrelevant *who study syntax.*

On the other hand, there is some flexibility in the distance, measured in dependencies, between the noun and this verb. For instance, it seems clear that *all* is the root of the phrase *all of the students who took syntax,* but it is *students,* not *all,* that is the antecedent of *who took syntax,* and when this is extraposed the parser has to look beyond the immediate dependents of the verb *passed:*

(39) All of the students passed *who took syntax.*

But it seems clear that the greater its distance from the verb, the less plausible a noun is as a potential antecedent, even when it is the only pragmatically sensible antecedent. Consider the following series:

(40) a I put the book down *that I was reading.*
b ?I wrote the name of the book down *that I was reading.*
c ?*I wrote the name of the author of the book down *that I was reading.*
d *I wrote the name of the agent of the author of the book down *that I was reading.*

It is especially interesting to notice in these examples how the determining factor must be the 'structural' distance, measured in dependency links, and not the 'physical' distance, measured in words, because the latter stays constant (*that* is always separated from *book* only by one word, *down).*

Another restriction on the antecedent noun is that it should be as recent as possible. It doesn't have to be the most recent of all, as witness examples like the following, where the most recent noun is of course *marks,* but the antecedent is *students:*

(41) Students generally get good marks *who take Advanced Word Grammar.*

On the other hand acceptability seems to decrease rapidly as the intervening nouns multiply or become more plausible antecedents.

In short, the antecedent has to be easy for the processor to find, and the easier it is, the more acceptable the sentence. It is very hard to see how these restrictions could be converted into clear grammatical constraints, but this may be the wrong way to approach them. Suppose we accept the Right-roof constraint as the only contribution that the grammar makes, so the antecedent of the extraposed adjunct

must be a noun which is subordinate to the same verb as the extraposee itself, and to no other verb. In principle this could leave a great deal of uncertainty for the parser, so we assume that speakers try to reduce this uncertainty, and hearers, in their role as judges of acceptability, rebel when the burden becomes intolerable. Uncertainty can be expected to correlate with such things as the distance between the extraposee and its antecedent, especially as measured in terms of dependencies.

It is instructive to compare the effects of extraposition with the other three examples that we have looked at. Subject-raising and tough-movement seem to cause no processing difficulties at all, because the 'displaced' element could be found by simply applying rules of the grammar. Wh-movement is a little harder to deal with because there is a limited element of choice, which produces some ambiguities (such as the famous example *Who do you want to succeed?),* but this uncertainty is so limited that extraction can apply more or less effortlessly over great structural distances. Both subject-raising and extraction are very common in all kinds of texts, and tough-movement is presumably as common as the adjectives that trigger it; in fact, all three constructions are more or less obligatory components of the constructions in which they occur.

In contrast, extraposition from NP is potentially difficult to process, because the parser has to explore a large section of the preceding clause rather than simply looking for some specific dependent of the head word. The speaker has to set these processing costs against the benefits (to the hearer) of delaying a long phrase to the end of the sentence, which explains the delicate shades of acceptability noted above. And, of course, a crucial difference between extraposition from NP and the other discontinuity-producing constructions is that it is completely optional.

9 Conclusions

The main point that I have tried to make in this paper is that dependency grammar provides at least as good a basis for analysing discontinuous phrases as does phrase-based grammar. What I see as the greatest advantage of DG in this respect is its ability to accommodate discontinuous phrases in a **single** structure, which respects the 'surface' facts of the observable words and their order as well as the more abstract relationships which (in some cases) lead to discontinuities. In this sense, then, DG, and Word Grammar in particular, is absolutely 'minimalist' in that it assumes only two units in syntactic structure: observable, ordered words, and unobservable and abstract relations between them. Neither of these assumptions is at all controversial, so if a theory assumes any more than this its supporters must show that the extra units are indispensible.

However I have also tried to approach syntactic analysis with one eye on the need to reconcile any grammatical analysis with a theory about how it could be applied by a parsing system (whether human or mechanical). The parser that I suggested is plausible, and demonstrably compatible with the fundamental theoretical claim of this paper, which is that the 'glue' that holds phrases together is the No-tangling Principle.

References

Blake, B. (1990) *Relational Grammar*. London: Routledge.

Bloomfield, L. (1933) *Language*. New York: Holt, Rinehart and Winston.

Bresnan, J. (ed.) (1982) *The Mental Representation of Grammatical Relations*. Cambridge, MA: MIT Press.

Butler, C. (1985) *Systemic Linguistics: Theory and practice*. London: Batsford.

Chomsky, N. (1957) *Syntactic Structures*. Amsterdam: Mouton.

Chomsky, N. (1982) *Lectures on Government and Binding.* Amsterdam: Foris.

Chomsky, N. (1986) *Barriers*. Cambridge, MA: MIT Press.

Covington, M. (1990) A dependency parser for variable-word-order languages. *Computational Linguistics 16*, 234-236.

Fraser, N. (1993) *Dependency Parsing*. London University PhD dissertation.

Fraser, N. and Hudson, R. (1992) Inheritance in Word Grammar. *Computational Linguistics 18*, 133-158.

Gaifman, H. (1965) Dependency systems and phrase structure systems. *Information and Control 8*, 304-337.

Gazdar, G.; Klein, E.; Pullum, G.; and Sag, I. (1986) *Generalized Phrase Structure Grammar*. Oxford: Blackwell.

Halliday, M. (1985) *An Introduction to Functional Grammar*. London: Arnold.

Hudson, R. (1984) *Word Grammar*. Oxford: Blackwell.

Hudson, R. (1988a) Coordination and grammatical relations. *Journal of Linguistics 24*, 303-342.

Hudson, R. (1988b) Extraction and grammatical relations. *Lingua 76*, 177-208.

Hudson, R. (1989a) Gapping and grammatical relations. *Journal of Linguistics 25*, 57-94.

Hudson, R. (1989b) Towards a computer-testable Word Grammar of English. *UCL Working Papers in Linguistics 1*, 321-339.

Hudson, R. (1990) *English Word Grammar*. Oxford: Blackwell.

Kunze, J. (1975) *Abhängigkeitsgrammatik*. Berlin: Akademie-Verlag.

Kunze, J. (ed.) (1982) *Automatische Analyse des Deutschen*. Berlin: Akademie-Verlag.

McCawley, J. (1988) *The Syntactic Phenomena of English*. Chicago: University of Chicago Press.

Mel'čuk, I. (1979) *Studies in Dependency Syntax*. Ann Arbor: Karoma.

Mel'čuk, I. (1988) *Dependency Syntax: Theory and practice*. Albany: State University of New York Press.

Pollard, C. and Sag, I. (1987) *Information-based Syntax and Semantics I: Fundamentals.* Stanford: CSLI.

Pulman, S. (1992) Parsing. In W. Bright (ed.) *International Encyclopedia of Linguistics, 3*, Oxford: Oxford University Press, 159-162.

Robinson, J. (1970) Dependency structures and transformational rules. *Language 46*, 259-285.

Sgall, P.; Hajičová, E.; and Panevová, J. (1986) *The Meaning of the Sentence in its Semantic and Pragmatic Aspects*. Prague: Academia.

Tesnière, L. (1959) *Eléments de Syntaxe Structurale*. Paris: Klincksieck.

Wood, M. (1993) *Categorial Grammar*. London: Routledge.