Parsing with Principles: On Constraining Derivations

HANS VAN DE KOOT

Abstract

In the recent parsing literature there have been relatively few attempts at giving a precise implementation of (some version of) Government Binding Theory. In this paper I discuss some approaches to principle-based parsing that aim at providing faithful implementations of Government Binding grammars. Apart from the fairly serious decidability problems facing simple-minded implementations, having more than one syntactic level can give rise to rather extreme inefficiency. This is because initial hypotheses about the structure of the input are insufficiently constrained at the level that is constructed first. I discuss some techniques that are generally applied to remedy this shortcoming and argue that these are largely unworkable because they do not address the problem head-on.

In the second part of the paper I argue that what is required to make GB-parsing feasible is a shift from constraints on representations to constraints on derivations. I reinterpret a few constraints on representation as constraints on derivation and demonstrate how this approach drastically reduces the vast search space associated with the standard formulation of the GB parsing problem.

1 Problems in computational modelling of GB-theory

1.1 Introduction

Of many informal proposals for GB-based parsers that have been put forward it is not clear what the relation is between the parser and the grammar that it is supposed to implement (see e.g. Abney 1986, Berwick 1987, Berwick & Weinberg 1985, Kashket 1987). Some of these parsers do not even embody all the knowledge characterized by the grammar: for instance, because they have no way of characterizing LF-representations. Although I think that building such toy parsers can give us some insight into the possibilities opened up by principle-based systems, they are not generally to be considered parsers for GB theory.

In this paper I will be concerned with what might (perhaps misleadingly) be dubbed "logic-based" approaches to GB-based parsing. Work in this area tends to be aimed at providing faithful implementations of current transformational grammars, often facing efficiency problems head-on, as we will see. I will limit myself here to a discussion of just three examples of this line of enquiry. First, I briefly discuss the work of Kolb & Thiersch (1990), Johnson (1989), and Stabler (1990a,1990b). The aim of this overview is to sketch the key computational problems encountered in faithfully implementing a multi-level principle-based grammar.

1.2 Kolb & Thiersch 1990: Polystratal vs. monostratal GB-parsing

Kolb & Thiersch (1990) discuss a major obstacle in designing a feasible parsing strategy for a multi-level theory like GB. When starting a parse, all one has to work on is the input string, roughly corresponding to Phonetic Form. But the only interface to the lexicon is located at D-Structure. This means, according to Kolb & Thiersch, that if one were to take the theory literally there is exactly one parsing strategy that the theory allows: one can generate well-formed strings starting at D-Structure until it produces one that matches the input. The disadvantages of such a generate-and test procedure are obvious: it is incredibly inefficient and produces a halting problem if the input is ungrammatical (because there are infinitely many possible D-structures).

Kolb & Thiersch then point out a much less obviously intractable strategy. The idea is to allow access to the lexicon at PF, using the lexical information on subcategorization and selection. In addition, PF would be allowed access to the X'-module. Note that this step implies the linguistically relevant decision that X'-theory applies not only at D-Structure. In this way one could generate all structures that are compatible with X'-theory and properties of lexical items, using modules at other levels to filter the bad ones out. This strategy introduces a new intermediate level of representation, call it *?-Structure*, consisting of a labelled bracketing possibly including empty categories, which is the source of a new halting problem. The point is that ?-Structure may contain empty categories, but the modules constraining the occurrence of empty categories are not available at this level. The authors point out that *ad hoc* restrictions do not help very much. For instance, if one allows gaps in argument positions only, the result is still an intractable number of possibilities. With Chomsky-adjunction allowed at any bar-level, but X'-theory restricted to binary branching, one gets more than 35000 different ?-structures for the simple German subordinate clause *...dass der Karl den Hund schlug* ("...that Karl beat the dog"). Kolb & Thiersch conclude from this:

> "Bad structures must be kept from being generated in the first place. But the generation of a faulty structure is only preventable, if all

> relevant conditions are checked as soon as possible during structure building, i.e. they must be reinterpreted as conditions on structure assignment.
>
> Aside from GB-theory, the equivalence between the declarative and the procedural view on structural conditions is, for all practical purposes, a straightforward fact. What complicates matters in our case, however, is that in the standard formulation of GB-theory the relevant conditions don't refer to the same levels, i.e. structures. Hence, a procedural reinterpretation of the modules alone isn't enough. The interconnection of the levels, and their contribution to the grammatical well-formedness of a sentence, also have to be captured in an incremental way."

It should not be difficult to guess what Kolb & Thiersch's next step is. They first observe that it is to date a matter of debate whether the mappings to derive S-Structure from D-Structure and LF from S-Structure are truly *derivational*, i.e. if there exists a mapping which is *itself* subject to certain constraints (e.g. subjacency or the ECP), where the constraints are too weak to define the levels by themselves, or if the levels are in fact *constituted* by sets of constraints. On the latter view, the constraints do not apply to the mapping itself but only to its output. Kolb & Thiersch consider the more declarative view of the mapping more congenial to parsing and, there being no conclusive evidence either way, they dismiss the derivational view. This sets the scene for a rather obvious second step: they collapse the theory into a single "annotated S-structure", which contains all the information of the original levels. Kolb & Thiersch then observe that this is a pseudo-alternative, since in a canonical S-structure all D-structure properties are trivially represented, except for the fact that every argument must be associated with exactly one θ-position (the θ-Criterion). One could, however, reformulate the θ-Criterion so that it holds at S-Structure. The level of LF could similarly be compiled into S-Structure, leaving us with a monostratal variant of the original polystratal theory.

While Kolb & Thiersch approach to GB-parsing is very much a common sense one, the kind of grammar compilation that they advocate should come with a guarantee that it is meaning preserving. It is in this respect that their proposal is clearly lacking. This lack of precision is all the more worrying, as the only researcher in the field who has tried to restructure GB-theory into a mono-stratal variant *and* prove logical equivalence has failed to do so (Stabler, p.c.).

1.3 The PAD parsers

1.3.1 Parsing as Deduction Johnson (1989) puts forward the "Parsing as Deduction" hypothesis: the human language processor uses its knowledge of a language to obtain knowledge of the utterances of that language in the same way that a theorem prover uses axioms to deduce their consequences. As Johnson points out, this hypothesis is "admittedly schematic, constituting more an outline for research than a hypothesis for direct empirical confirmation, and is intended to be construed as a refinement, rather than a rejection, of other currently proposed processing models" (p.106).

Johnson's hypothesis is not new, as he himself intimates by referring to Pereira and Warren's (1983) paper "Parsing as Deduction". There is, however, one aspect of the Parsing as Deduction (PAD) hypothesis that deserves special attention:

> "Viewing parsers as highly specialized theorem provers naturally leads one to distinguish the knowledge of language used by the parser (i.e. the hypotheses and inference rules) from the manner in which this knowledge is put to use (i.e. the inference control strategy which directs the application of the inference rules)." (pp.106-107)

In other words, it allows us to experiment with the inference control strategy, while keeping the grammar constant. Most of Johnson's paper is concerned with exactly this aspect of his hypothesis and I will return to it below.

A second point Johnson addresses, and which he considers "one of the major conceptual shifts embodied in the deductive approach" (p.107), is the tendency for deductive systems to move away from *representation-oriented* models of processing to *knowledge-oriented* models. To see what he has in mind, recall that GB theory posits four levels of representation: D-Structure, S-Structure, PF (phonetic form) and LF (logical form). A representation-oriented parser would construct explicit representations for each of these four levels. A knowledge-based parser, by contrast, need not necessarily build all these representations. The point is simply, says Johnson, that the use of knowledge of a particular level of representation does not entail the explicit construction of that representation. He demonstrates this by exhibiting a PAD parser which uses knowledge of D-Structure in the parsing process, yet avoids the explicit construction of this level of representation. We will see later on how this issue is connected with grammar compilation[1].

---

[1]It is a common practice to translate (or *compile*) the grammar to be used into a form that speeds up the parsing process.

1.3.2 The logical structure of a PAD parser According to the PAD Hypothesis, a parser is just a special purpose inference procedure computing over the theory of Universal Grammar. "To construct a deductive parser for GB one builds a specialized theorem prover for GB theory, provides it with parameter settings and a lexicon as hypotheses, and uses it to derive the consequences of these hypotheses that describe the utterances of interest" (p.115).

A PAD parser is constructed using a Horn-clause theorem prover (a Prolog interpreter). The theorem prover is provided with three sets of clauses: (i) an axiomatization of the theory of Universal Grammar, (ii) the parameter values for the language to be parsed, and (iii) a lexicon. Ideally the axiomatization reflects the internal organization of the principles of the grammar (here GB). In fact Johnson's axiomatization, to the extent that he discusses it, does not always meet this desideratum.

1.3.3 Control strategies Johnson's first parser PAD1 has all the bad characteristics of a blind generate-and-test procedure[2]. The way it goes about parsing a sentence is like this. First, it generates all D-structures that satisfy X'-theory and filters all those that fail to satisfy θ-theory[3]. It then computes all corresponding S-structures using move α and removes all S-structures that fail to satisfy the Case Filter. It is not until this point that it determines if the terminal string of the S-structure is identical to the input string. Clearly, the PAD1 parser is incredibly inefficient and it should not be very difficult to do better, as indeed it is not.

PAD2 uses extensive co-routining between the different modules of the grammar. It does this by checking each existing node for well-formedness with respect to the principles of grammar and by verifying that the terminal string of the partially constructed S-structure matches the input string before it creates any additional structure. PAD2 is textually identical to PAD1, but achieves its improved performance through a variant of the *freeze* predicate of Prolog II. The control strategy used in PAD2 allows inferences for certain predicates to be delayed until specified arguments to these predicates are (partially) instantiated. When some other application of an inference rule instantiates such an argument, the current sequence of inferences is suspended and the delayed

---

[2]PAD1 uses the SLD inference control strategy of Prolog. In combination with Johnson's (very straightforward) axiomatization, this produces a simple-minded generate-and-test parser.

[3]If X'-theory admits infinitely many D-structures, then the resulting generate-and-test procedure is not guaranteed to terminate on ungrammatical input. For this reason, Johnson uses a modified version of X'-theory in his PAD1 parser that allows only finitely many D-structures.

inferences take precedence. Johnson describes the effects of this control strategy as follows:

> "Inferences using the X', θ, Case, Move α and LF-movement principles are immediately delayed since the relevant structures are uninstantiated. The "phonology" principle (a simple recursive tree-walking predicate that collects terminal items) is not delayed, so the parser begins performing inferences associated with it. These instantiate the top node of the S-structure, so the delayed inferences from the Case Filter, Move α and LF-movement are performed. The inferences associated with Move α result in the instantiation of the top node(s) of the D-structure, and hence the delayed inferences associated with the X' and θ principles are also performed. Only after all of the principles have applied to the S-structure node instantiated by the "phonology" relation and the corresponding D-structure node(s) instantiated by Move α are any further inferences associated with the "phonology" relation performed, causing the instantiation of further S-structure nodes and the repetition of the cycle of activation and delaying."
> (pp.119-120)

PAD2, like PAD1, is still top-down and left-to-right. But unlike PAD1, it builds D-structure, S-structure, and LF representations simultaneously.

1.3.4 Using knowledge vs. constructing representations Most work in computational linguistics tacitly assumes that a parser only implements a grammar if it builds all the representations specified by that grammar. As Johnson correctly points out, this is by no means a logically necessary position. To some extent, the point he raises is quite uncontroversial, as he acknowledges, since many standard parsing algorithms do not explicitly construct any syntactic representation at all. In fact, the cubic time complexity of the Earley algorithm relies on this. Some context-free strings are $O(e^n)$-ways ambiguous and hence no parser that had to explicitly construct all the possible syntactic structures for such a string could run in cubic time.

As it turns out in the case of the PAD parsers, some amount of grammar compilation leads to an axiomatization of UG that makes it possible to avoid the explicit construction of D-Structure and S-Structure. Johnson only gives an informal account of how this result is achieved. The first step is to observe that several predicates of the original axiomatization perform exactly the same recursive traversal of the same structure. Using the Unfold/Fold program transformation technique[4], it is possible to replace the X', θ, move α, Case and Phonology predicates by a single predicate "p" that holds of exactly the D-structure, S-structure, PF triples admitted by the conjunction of the original

[4]The Unfold/Fold transformation produces a logically equivalent program.

principles. This transformation maps PAD2 into the more efficient PAD3, which shows a drastic reduction in the amount of tree-walking that has to be done. Because it replaces the original axiomatization with a provably equivalent one, the PAD3 parser provably infers the same knowledge of language as the PAD1 and PAD2 parsers. Collapsing the above mentioned principles into the predicate "p" also has the effect that PAD3 exhibits the co-routining behaviour of PAD2, even when used with the standard SLD inference control strategy of Prolog.

PAD4 is quite like PAD3, except that it does not construct D-structures. As Johnson points out, no predicate ever uses the D-structure value returned by the predicate "p", not even "p" in recursive calls to itself. Therefore, "p" can be replaced by the predicate "p1", which is like "p" except that it does not have a D-structure argument.

PAD5 is constructed from PAD2 by applying the Unfold/Fold transformation to all principles of grammar simultaneously. This yields an axiomatization in which both D-structure and S-structure values are always ignored and hence deletable. Thus PAD5 only constructs an LF representation for its input.

1.3.5 Discussion Although Johnson's work seems quite promising at first sight, it becomes much less so when one considers his claims in some more detail. I have already commented upon the triviality of some aspects of the PAD hypothesis. Here I will focus on a prime feature of his approach: the shift from representation-oriented models to knowledge-oriented models.

To some extent the application of the Unfold/Fold transformation to a logic program is a standard programming technique that sophisticated programmers often use without being aware of. However, it is quite a different matter to use this technique to improve the efficiency of a large and possibly complicated logic program. In the case of Johnson's PAD parsers the success of the transformation would seem to be a rather accidental by-product of the details of the implementation. It would seem therefore that the Unfold/Fold transformation is not the sort of general method that is going to make a GB-parser efficient. This is not to say that the opposition between representation-oriented and knowledge-oriented models is not, in itself, an interesting one. Johnson's formalization of (part of) GB-theory seems to be quite faithful and, as the Unfold/Fold transformation is meaning-preserving, the same is true of the PAD parsers that do not build all representational levels explicitly.

In this context it should also be stressed that the PAD parsers only parse a very restricted fragment of English, which does not even include an account of *wh*-movement. Johnson claims that this should not worry us to much, as "the techniques used to construct these parsers are quite general, and there appears to be no reason that they would not extend to a more substantial fragment". My impression is that this is a bit too optimistic. *Wh*-movement complicates the parsing problem considerably, because it dramatically increases the parser's search space. As a result of such movements, the leftmost constituent in an S-structure can be located arbitrarily deep and to the right in its corresponding D-structure. This in combination with the fact that lexical insertion and X'-theory are enforced

at D-Structure gives rise to severe thrashing behaviour[5]. It is by no means clear if and how transformation techniques could come to the rescue here. See also section 1.4.1 below.

On the positive side of things, Johnson's approach highlights the relevance of experimenting with various control strategies. It is quite clear, even from his rather sketchy presentation, that co-routining between the different modules of the grammar eliminates a substantial portion of wasteful search.

1.4 Stabler (1990a,b): reformulating the GB parsing problem

1.4.1 Identifying the major sources of intractability Formalizing a GB grammar in first-order logic is not too difficult, as the examples below (from Stabler 1990b) illustrate. For instance, the well-formedness of D-Structure, S-Structure, and Logical Form can be expressed as follows:

(1) $(\forall T)\ d\_structure(T) \leftrightarrow x\_bar(T) \wedge lexical\_insertion(T) \wedge \theta\text{-}marking(T)$
(2) $(\forall T)\ s\_structure(T) \leftrightarrow case\_theory(T)$
(3) $(\forall T)\ lf\_structure(T) \leftrightarrow ecp(T) \wedge binding\_theory(T)$

We also have to define under what conditions two levels of representations may be taken to be related by zero or more applications of move α. Following the standard definition of this transformational rule, move α is either a substitution or an adjunction. This is expressed in (4). The logical sentence (5) gives the transitive closure of the *move_α* relation *move_α**.

(4) $(\forall T_0,T_1)\ move\_\alpha(T_0,T_1) \leftrightarrow substitute(T_0,T_1) \vee adjoin(T_0,T_1)$
(5) $(\forall T_0,T_1)\ move\_\alpha^*(T_0,T_1) \leftrightarrow (T_0 = T_1 \vee (\exists T_2)\ move\_\alpha(T_0,T_2) \wedge move\_\alpha^*(T_2,T_1)))$

Now suppose we think of the parsing problem as providing a constructive proof, given an input *pf*, for (6):

(6) $\exists DS,SS,LF$ $\quad d\_structure(DS) \wedge move\_\alpha^*(DS,SS) \wedge s\_structure(SS) \wedge lf\_movement^*(SS,LF) \wedge yield(SS,pf)$.

The predicate *yield* simply gives the sequence of leaves of *SS* (i.e. we abstract away from the phonological component). As we discussed earlier, simple methods (generate-and-test, backtracking) will be ridiculously inefficient. As Stabler (1990b) points out the backtracking procedures that use extensive co-routining

[5]Repeatedly considering and rejecting the same partial, incorrect solutions.

(such as Johnson's 1988 PAD2-5 parsers) run into two sorts of difficulties: (i) as a result of the application of move α, the leftmost constituent in an S-structure can be located arbitrarily deep and right in the corresponding D-structure, (ii) lexical insertion and X'-theory are enforced at D-Structure[6]. He explains:

> "The problem is that the input string is the yield (PF) of an S-structure, but basic X-bar constraints and lexically specified information about the elements of the string cannot be used to restrict the search for a proof until an S-structure (at least partially formulated) has been related by some particular number of movement relations to a D-structure (at least partially formulated) containing those elements. Since we do not know in advance how many movements may have occurred or where in the D-structure the leftmost S-structure constituent might have originated, building the leftmost branches or any other particular part of the D-structure first will not generally suffice to specify the leftmost constituent in the corresponding S-structures. Without this specification, the search will be wasteful, repeatedly considering and rejecting bad analyses."
> (p.5)

Stabler (1990b) gives several examples of how *the grammar itself* reveals that a great deal of search done by e.g. the simple generate-and-test procedure is useless. He exhibits a number of principles of induction, some of which allow one to import certain constraints on D-Structure into the level of S-Structure and to add constraints on the mappings between levels. Perhaps paradoxically, imposing extra constraints on the parsing problem helps to reduce the search space. I say paradoxically, because the vast search space of the GB parsing problem is a direct result of parsing with a set of very general constraints. As we will see, however, the point of Stabler's method is to add the new constraints to the parsing problem itself (i.e. to (6) above) rather than to the grammar. If the new constraints are relatively easy to evaluate, then a selective backtracking approach which interleaves the old and the new constraints to maximum advantage can lead to a considerable reduction in search space.

1.4.2 Using the grammar to reformulate the parsing problem Stabler uses two inductive principles from which he derives several new constraints that help to alleviate the parsing problem. Let us consider each of these in turn. The first of these principles says that for any property φ, if φ is preserved by one movement, then it is preserved by any number of movements. More formally, we have (7)-(10), where moveα^* is the transitive closure of *move*α:

[6]Stabler makes the conservative assumption that D-Structure cannot be dispensed with.

(7) $(\forall T_0)\ d\_structure(T_0) \rightarrow \phi(T_0)$

(8) $(\forall T_0,T_1)\ \phi(T_0) \rightarrow (move\alpha(T_0,T_1) \rightarrow \phi(T_1)$

(9) $[(\forall T_0,T_1)\ \phi(T_0) \rightarrow (move\alpha(T_0,T_1) \rightarrow \phi(T_1)] \rightarrow$
$[(\forall T_0,T)\ \phi(T_0) \rightarrow (move\alpha^*(T_0,T) \rightarrow \phi(T))]$

---

(10) $(\forall T_0,T)\ d\_structure(T_0) \rightarrow \phi(T_0) \rightarrow (move\alpha^*(T_0,T) \rightarrow \phi(T))$

Suppose ϕ is *lexical_insertion*, then (7) follows immediately. If the only movements we consider are substitution and adjunction of maximal projection, then movement will never change anything inside a word and (8) holds as well. Premise (9) is a straightforward principle of induction, quite analogous to the principles that license inductive arguments in mathematics. The conclusion (10) clearly follows from the premises. So we conclude that lexical insertion conditions hold at S-Structure (on the assumptions about movement just mentioned). There are quite a few D-Structure properties for which we could construct a similar argument. Following standard linguistic terminology, Stabler calls principles like these *structure preserving principles*.

Things become somewhat more complicated if one wants to demonstrate that certain relations hold between levels. Stabler mentions the following example. If an NP with certain features occurs at D-structure, it must occur in every corresponding S-structure, and, conversely, if a nonempty NP node with certain features occurs at S-structure, then an NP node with those features occurs in every corresponding D-structure. This relation of having XPs in common could be defined with an axiom like the following:

(11) $(\forall T_0,T)\ same\_xps(\forall T_0,T) \leftrightarrow (\forall XP)(occurs(XP,T_0) \rightarrow occurs(XP,T))$

The corresponding facts we need would look like (12):

(12) $(\forall T_0,T)\ move\alpha^*(T_0,T) \rightarrow same\_xps(T_0,T)$

Establishing this result is not as straightforward as one might think. The trouble is that the mere fact that trees related by *move*α always stand in relation ψ does not entail that trees related by *move*α^* stand in relation ψ. This can easily be demonstrated. Suppose ψ is the relation that holds between two trees just in case the second has exactly one more trace than the first. This relation holds whenever *move*α holds, but it does not necessarily hold whenever *move*α^* holds. Stabler therefore uses a somewhat complex argument to deliver the required results:

(13) $(\forall T)\ \psi(T,T)$

(14) $(\forall T_0,T_1,T)\ (\psi(T_0,T_1) \wedge move\alpha(T_1,T)) \rightarrow \psi(T_0,T)$

(15) $[(\forall T_0,T_1,T)\ (\psi(T_0,T_0) \wedge ((\psi(T_0,T_1) \wedge move\alpha(T_1,T)) \rightarrow \psi(T_0,T)] \rightarrow$
$[(\forall T_0,T)\ move\alpha^*(T_0,T)) \rightarrow \psi(T_0,T)]$

---

(16) $(\forall T_0,T)\ move\alpha^*(T_0,T)) \rightarrow \psi(T_0,T)$

In other words, if a relation ψ is reflexive and is preserved by *move*α, then it is preserved by *move*α*. He dubs principles of this form *relation preserving principles*. Let us now consider how facts derived with the help of the induction principles can be put to use.

1.4.3 Gaining efficiency by adding lemmas to the parsing problem Stabler points out that, in principle at least, there are three ways of putting the inductive Principles to use and each of these implies a different view of the runtime grammar the parser employs.

First, we could go on parsing with the grammar that we had and try to deduce valuable restrictions on-line. This is not a feasible approach, however, for a number of reasons. There are a lot of properties and relations that are preserved by movements, but not all of these are equally useful. We are looking for cases that have relatively few provable instances. Figuring out which properties and relations are useful is not an easy matter. A second problem that is perhaps even more distressing is that there are infinitely many principles of induction and they can be arbitrarily complex. This means that even if we know in advance which structure preserving principle we want to establish, a proof from the grammar may be very hard to find.

This leads us to consider a second option: we could find interesting cases of structure preserving principles off-line and add them as lemmas to the grammar. Of course, since they are logical consequences of the grammar, these additions will preserve logical equivalence. According to Stabler, there are some problems with this approach. Lemma generation introduces a lot of redundancy and as a result the search space will become larger rather than smaller. Also there will be many ways to prove the same result, although some of the new proofs will be shorter than any of those previously available. Stabler observes that lemma generation method may be of psychological interest if language is richly redundant in the way that Minsky and others believe most human reasoning is, but he finally settles on a third option that offers more hope of a reduction in search space.

The third possibility is to add the lemmas to the parsing problem. This will transform the original parsing problem (6) into (17) below:

(17) $$\exists DS,SS,LF \quad \begin{array}{l} d\_structure(DS) \wedge \\ move\_\alpha^*(DS,SS) \wedge \\ s\_structure(SS) \wedge \\ lf\_movement^*(SS,LF) \wedge \\ yield(SS,pf) \wedge \\ \phi_1(SS) \wedge \ldots \wedge \phi_n(SS) \wedge \\ \psi_1(DS,SS) \wedge \ldots \wedge \psi_i(DS,SS) \wedge \\ \psi_{i+1}(SS,LF) \wedge \ldots \wedge \psi_m(SS,LF) \wedge. \end{array}$$

What makes this strategy effective is that selective backtracking will allow us to interleave the steps of each subproblem to obtain the maximum advantage. The idea is to test partial evaluations of the original problem by interleaving them with evaluation of the lemmas. In this way incorrect solutions can be discovered at a

very early stage. How cheap the reduction in search will be depends on how easy to evaluate the lemmas are. Stabler points out that problem solving methods of this kind are sometimes called "relaxation" methods in loose analogy with the better known iterative probabilistic relaxation methods. The lemmas that are added to the original problem can be regarded as defining relaxed, approximate solutions which can be found and then tested against the original constraints. The technique is similar in motivation to the consistency methods of Mackworth (1977,1987) and others, but it differs from those in that the original constraints of the problem remain unchanged.

1.4.4 Discussion Of the approaches to GB parsing that I have discussed Stabler's is the most sophisticated, both in terms of logical precision and linguistic accuracy. In my view, this makes his work particularly attractive. Still, there are some questions that deserve our attention. It would seem that, while addition of lemmas to the parsing problem rather than to the grammar avoids introducing redundancy in the grammar, it does give rise to redundancy in the proofs. This is because the lemmas that are evaluated for S-Structure are reevaluated for D-Structure. There may be ways of getting rid of such redundancies by eliminating the D-Structure constraints that constitute the source for the lemmas that one deduces. This would make Stabler's proposal much closer in spirit to Kolb & Thiersch's. Indeed, one might well ask whether it would not be more sensible to include the lemmas as S-Structure constraints in the grammar, while at the same time removing the D-Structure constraints from which they were derived. This would result in a greatly impoverished D-Structure. On this view it is largely an empirical matter whether or not D-structure can be eliminated from the grammar altogether. It might just be the case that certain linguistically well-motivated D-Structure constraints cannot be reformulated so as to hold at any level other than D-Structure.

1.5 Conclusion

I have discussed three rather different attempts to address the problem of developing a feasible GB-parser. One important observation about these logic-based approaches is that there seems to be a *communis opinio* what it is about GB-theory that makes its parsing problem exceptionally hard. GB-theory postulates several levels of representation, each of which is associated with a number of constraints, which gives rise to potential undecidability and may cause gross inefficiency. This is because initial hypotheses about the structure of the input are insufficiently constrained at the level that is constructed first. Even though the three approaches I have discussed differ in non-trivial ways, there is a key property that they share: each proposal is an attempt to enrich the set of constraints that can be brought to bear on the initial hypothesis about the input. This is achieved in different ways. Kolb & Thiersh "collapse" the theory into one level, Johnson's main weapon is co-routining, and Stabler uses the lemmas he derives from his inductive proofs to relax the parsing problem. This relaxation

technique has the effect of bringing D-Structure and LF constraints to bear on S-Structure.

Given that the three logic-based approaches are similar in the respects just mentioned, one might well ask if this similarity is accidental or a consequence of the unavailability of alternatives.

The following section is an attempt on my part to show that there are alternative ways of reducing the search space of the GB parsing problem. The idea that I will explore emphasizes the importance of constraining the *mapping* between levels of representation and accords a secondary role to grammar compilation aimed at enriching the set of constraints at each of the levels of representation. The proposals discussed above share the characteristic that they use the constraints that govern the form of chains of movement as *filters*. This is by no means necessary or even desirable, but it is a natural consequence of taking a *declarative* view of the rule move α. Taking the filters approach to its logical extreme has the obvious drawback that one is left without any means of reducing the search space associated with the *move*_α* relation, except through varying the inference control strategy. It therefore seems worthwhile to explore the alternative, *derivational* view of the rule move α. On this view of the rule, constraints on the form of chains should be interpreted as constraints on *derivations*, wherever this is possible.

2 An Informed Generate-and-Test Method

2.1 Decidability at D-Structure

The suggestion and informal result that I discuss immediately below are at once boringly trivial and very useful. Let us return to the naive generate-and-test method and the cause of its decidability problem. In a nutshell what is going wrong is that the parser can keep on guessing D-structures for ungrammatical inputs ad nauseam. As a matter of fact, one might have to wait a very long time until it guessed the correct D-structure for a *grammatical* input, which is why nobody would ever seriously consider this method anyway.

The assumption in the literature I have discussed is that the only way of testing D-structures against the input string is at PF. Given this assumption and assuming co-routining is a clever idea by anyone's book, there are exactly two options left. One is generate-and-test, which gives rise to decidability problems, no matter how cleverly one co-routines the construction of DS, SS, and PF. The other is the bottom-up approach, which can be achieved by freezing the appropriate predicates at the appropriate levels. The bottom-up approach in its "pure" form (i.e. without grammar compilation of any kind) gives decidability problems as well. Here the problem is that one can keep on guessing S-structures for ungrammatical inputs by postulating more and more traces at S-structure. We have seen what the standard lines of attack are. Either one tries to get rid of one or more levels of representation by extensive compilation, or one uses slightly less drastic measures, which still have the effect, however, of bringing D-Structure and/or LF constraints to bear on S-Structure or PF. While such strategies may very

well turn out to be unavoidable, it seems worthwhile to investigate possibilities which leave the grammar as unaffected as possible.

One such possibility lies in dropping the assumption that the input string is only available at PF[7]. One might well ask what motivates this assumption in the first place. Although it is certainly true that lexical items may be added and deleted in the course of a derivation, deletion and insertion rules certainly do not affect the mappings between levels completely randomly. Deletion takes place upto recoverability and insertion is restricted to designated elements. Let us assume, here without argument, that these discrepancies between D-structures and PFs can be dealt with one way or another. It is now of course entirely straightforward how we can use the input to constrain the generate-and-test method in such a way as to eliminate the undecidability that rendered the *blind* generate-and-test method useless.

The argument takes on the following general form: given an input string σ, there exists a maximal DS δ^{8}, such that if there is no valid derivation for σ from δ, then there is no valid derivation for σ from any DS[9]. Let us now consider the argument in some more detail. Given an input string, what is it that determines the upper bound on the size of its DS? Clearly the upper bound on the size of a DS is determined by the number of verbs in the input. Simply put, any main verb will license the projection of *one* VP and its associated functional projections (i.e. IP and CP). Since functional projections may have empty heads (possibly even at DS), we seem to be in for trouble. This is not a real problem, however, because functional projections only recurse through lexical projections. As a result, the number of functional projections that could be hypothesized for a given input is linear in the number of lexical projections that are hypothesized, and all is well. All we really need to establish, then, is whether the number of empty *lexical* heads that could be hypothesized for any given input is always finite. It is here that recoverability comes to the rescue: the occurrence of null heads is subject to very stringent restrictions. Null heads need to be locally licensed in one way or another.

Notice that we need not worry at all about the possibility of having to hypothesize empty NPs at D-Structure. Why is that? First, there are no traces at D-Structure. Second, there can be empty pronominal elements, like PRO and pro, at D-Structure, but like any other NP such empty pronominals must satisfy the θ-Criterion. This has the immediate consequence that their number is linearly related to the number of selecting heads, and all is well again.

Returning now to the main line of argument, it seems fairly straightforward to maintain the position that the number of verbs in an input string σ will determine an upper bound β on the size of DS δ. This is because the number of

---

[7]We continue to abstract from phonological processes, so that a PF-representation is just the sequence of leaves of SS.

[8]Where the size of a DS may be simply expressed in terms of the number of nodes it contains.

[9]This is not quite accurate, since there may be more than one maximal DS.

verbs in the input (roughly) determines the number of dependent projections. As a result, we know what size the tree has to have for the input to be able to satisfy the θ-Criterion. It follows that, if there is no valid derivation for a string σ from any DS of size ß or less, then there is no valid derivation for σ at all. We therefore (informally) conclude that the parsing problem for this informed version of generate-and-test is not undecidable "at D-Structure". This is not to say that it could not be undecidable for other reasons.

2.2 Decidability at S-Structure

2.2.1 The mapping problem We have established that the parsing problem for our generate-and-test procedure is not undecidable because of properties of D-Structure, but can we establish a more general result? There are good reasons to assume that, unless special restrictions are imposed on move α, the parsing problem for a GB-based parser is not generally going to be decidable. There is a very simple reason for this: multiple adjunction. If it is always possible to try one more adjunction, then clearly the parser is never going to report failure on ungrammatical input. This problem will arise regardless of whether one is using a generate-and-test or a bottom-up procedure. It is not easy to see how one can remove this shortcoming in a principled way. The problem does not just arise in the mapping from D-Structure to S-Structure, but in the mapping from S-Structure to LF as well. It is very unlikely that adjunction can be eliminated from the theory, while preserving logical equivalence. So we will have to find a way to get rid of undesirable multiple adjunctions. Note that we do not want to rule out the possibility of multiple adjunction altogether: we do want to allow *different* categories to adjoin to the same projection. Thus, the structure in (18) is well-formed:

(18) $[\text{to whom}]_k$ did you $[t_k''[\text{wonder } [_{CP} \text{what}_i \text{ to } [t_k'[t_i'[\text{give } t_i\ t_k]]]]]]$

It seems to me that there are two ways in which one could achieve the desired result. The first of these is simple: state the move_α* relation in such a way that adjunction of a given node cannot cannot be applied twice to any node[10]. A second approach one could explore is whether it is possible to constrain the rule move α in such a way that the move_α* relation yields all and only the correct outputs *without backtracking*. It seems likely that such a solution would also be very desirable from the point of view of efficiency. It is not at all obvious, however, that this can be done. It certainly cannot be done without either modifying the theory or extensive grammar compilation, as I will show in what follows. It is important to bear in mind that we can always resort to the stipulation

[10]Note that it is not enough to forbid *recursion* of move α on any node, as we also want to forbid A to readjoin to B even if some other node C has adjoined to B in the mean time.

that adjunction of a given node cannot occur twice to any node. Constraining the rule move α to less than full determinism will still greatly simplify the GB parsing problem. We investigate the reasons for this next.

2.2.2 Move α as a source of intractability Consider once more the general form of the GB parsing problem:

(19) $\exists DS,SS,LF \quad d\_structure(DS) \wedge move\_\alpha^*(DS,SS) \wedge s\_structure(SS) \wedge lf\_movement^*(SS,LF) \wedge yield(SS,pf).$

If we abstract away from the possibility of co-routining for the time being, it is easy to see that the *move_α** relation will be the source of massive backtracking. move α is a "blind" rule. Its formulation does not embody any constraints at all, not even constraints that determine what is a possible landing site. As a direct result of this, the *move_α** relation is associated with a vast search space. Of the constraints that filter out illicit movements some apply at S-Structure (e.g. Subjacency) and others at LF (e.g. ECP). The problem is exacerbated by the sheer number of potential landing sites in any tree of a reasonable size. Suppose we have a case of *wh*-movement in a tree with *n* nodes. Suppose further more that we only consider potential adjunction sites and disregard the possibility of multiple adjunction to the same node. Then, if we apply move α only once, there will be roughly *n-s* potential landing sites (where *s* is the number of nodes in the moved constituent). If we apply move α twice to the same node, there are $(n\text{-}s)(n\text{-}(s+1)$ possible derivations, and so on. In general, for a tree of size *n* (where *n* is the number of nodes, there are roughly

$$\sum_{k=0}^{n} \frac{n!}{(n-k)!}$$

possible derivations *for any node in the tree*, which is a fairly astronomical number for a tree of reasonable size[11].

[11]To get at least a vague idea of how rapidly this function grows, consider the table below:

n	f(*n*)
10	$9.86\ 10^{5}$
15	$3.55\ 10^{12}$
20	$6.61\ 10^{18}$
25	$4.21\ 10^{25}$

In practice, we can alleviate the search problem considerably by co-routining the construction of DS, SS and LF. However, while the use of such techniques is certainly not without merit, it leaves the basic problem unaffected: it does not reduce the search space associated with the *move*_α* relation itself. To achieve that, we would have to find ways of constraining the *application* of move α rather than its output.

Let me give a simple example. It does not seem very difficult to redefine the *move*_α* relation in such a way that a maximal projection can only be adjoined to a maximal projection. More generally, we could require that a node with bar level *k* can only adjoin to a node with bar level *k*. Just this simple modification will bring down *n* in the formula above by a factor 3, by no means a small reduction in search space, as the reduction is within the scope of the factorial[12].

In short, it seems worthwhile to explore the possibility of reducing the search space introduced by the *move*_α* relation, as this will certainly reduce the complexity of the overall parsing problem. But how should we go about doing this? One possibility is to compile chain-related constraints holding at S-Structure and LF into the *move*_α* relation. While this may seem quite attractive from a parsing point of view, it certainly runs counter to received wisdom in the linguistic community. This need not in itself be a problem. There are no a priori reasons to expect a very close relationship between the grammatical theory and its realization in a parser[13]. This is not to say, however, that we would not prefer the relation between a grammar and its implementation to be as simple as possible. In light of this, the contrast between the declarative and derivational view of move α takes on a new dimension. If one takes a derivational view of move α, one is more easily led to interpret a constraint like Subjacency as a condition on the application of this rule than as a condition on its output. That is, one is more inclined to view Subjacency as a constraint governing the *mapping* between D-Structure and S-Structure.than as a constraint holding at S-Structure. Even though linguists currently working in the field of GB-theory are divided on this matter, much of the theory is to a large extent open to either interpretation. The derivational view is in many ways more congenial to the approach to GB-parsing that I will advocate here. This is because the derivational interpretation of move α implies a natural division between chain-related constraints (i.e. constraints on movement) and constraints on representation. My proposal singles out the chain-related constraints as candidates for compilation into the *move*_α* relation.

---

[12]In other words, there is little reason for joy if n! is brought down to 1/3(n!). Things start to look a bit brighter if n! is reduced to (1/3n)!, as is the case here.

[13]I refer the interested reader to Marr (1982), Berwick & Weinberg (1984), van de Koot (1990) and Ristad (1990) for extensive discussion of this important but often neglected point.

2.2.3 Constraining the DS-to-SS mapping The constraints governing the distribution of traces are doing two rather different jobs. On the one hand, traces must satisfy certain locality requirements (Subjacency and the ECP)[14]. Sentence (20), for example, is a standard ECP violation:

(20) * John$_i$ seems that it is certain [t_i to win]

On the other hand, there are conditions that block so-called "improper" movement. Consider (21) *with the structure indicated*:

(21) * [$_{IP}$ John$_i$ [$_{VP}$ t_i [$_{VP}$ seems [$_{CP}$ t_i [$_{IP}$ Mary [$_{VP}$ t_i [$_{VP}$ saw t_i]]]]]]]

This structure does not violate Subjacency or the ECP. All movement is strictly local. Still, it is ill-formed. But how does the theory rule it out? The answer is that the structure violates the Binding Theory. The trace immediately following *saw* is an R-expression and therefore subject to condition C of the Binding Theory. By condition C this trace must be A-free. It is A-bound by *John*, however.

Where do these principles apply in the grammar? Subjacency is standardly assumed to hold at S-Structure. It is fairly easy to see, however, how it could be reinterpreted as a condition on the *mapping* from D-Structure to S-Structure. From our point of view this is a bonus, because it would allow us to constrain the derivation from D-Structure to S-Structure by compiling Subjacency into the *move*_α* predicate. Similar optimistic conclusions are probably not warranted in the case of the Binding Theory or the ECP, which are both assumed to hold at LF. But let us see how far we can get.

First, consider the ECP. Recall that the ECP, as formulated in Barriers, is actually "supported" by several processes, some of which (viz. those at LF) are ordered with respect to each other:

(22)
(i)	move α
(ii)	γ-marking (at SS, for non-adjunct traces output by (i))
(iii)	*wh*-raising
(iv)	free deletion of traces
(v)	γ-marking (at LF, for all traces not covered by (ii))
(vi)	γ-checking (i.e. the ECP)

Intuitively, the thing to do would be to reformulate γ-marking in such a way that it would act as a constraint on derivation. That is, it would be nice if we could compile γ-marking into the move_α* relation. It is fairly easy to see that this will not help much, however. This is because not *all* S-structure traces are subject to γ-marking, only the non-adjunct traces are. This will still leave us with a subset

[14] As a matter of fact the ECP is usually taken to consist of a locality requirement *and* an identification requirement. The present discussion focusses on the locality requirements that chain links must meet.

of the move α mapping that is not constrained until LF. There are several ways in which one might attempt to resolve this problem. Perhaps the most attractive of these is to investigate the consequences of the conjecture that there is no adjunct-movement in the mapping from D-Structure to S-Structure at all. Instead, one would have to assume that *all* chains of adjunct movement are constructed at LF. As noted in van de Koot (1990) the idea has some intuitive appeal, because adjuncts are not θ-marked and as a result their position is not projected from the lexicon. I will not investigate this possibility any further here.

What about the Binding Theory? First, observe that of the three binding conditions only Condition C is needed to constrain the form of chains[15]. The standard way of formulating this condition is as in (23):

(23) An R-expression must be A-free up to the domain of its associated operator

(23) suffices to rule out ill-formed sentences like (21) above. We understand (23) as follows: an R-expression must be A-free in the domain of its operator if it has one, otherwise it must be A-free throughout. There is good evidence that, if Condition C holds, it must be stated at LF. Given the approach I have been pushing in this section so far, this is a serious problem. It would be very encouraging if there were a neat division between constraints on representations and constraints on derivations. This is not what we find here, however. Although we could perhaps reformulate move α in such a way that at each successive step it observed Condition C, there is no way we could simultaneously eliminate Condition C at LF.

Brody's (1990) work on Case Theory and argumenthood would seem to offer good prospects for resolving the tension just noted. This work is of a rather fundamental nature and discussing it in great detail would certainly take us too far afield. I will therefore just sketch the problem he addresses and the implications of the solution that he comes up with.

2.2.4 Chain uniformity In standard GB-theory there are two notions of "argument": what is an argument at D-Structure is not always an argument at LF. This is clearly a problem. How does it arise? First, it is clear that if D-Structure exists, then the θ-Criterion should hold at this level: it is here that thematic properties are determined. If we look at NP-movement, then we see that having the θ-Criterion at D-Structure will simplify matters: at this level NP-movement has

[15]Not every GB-linguist would agree with this. There has been a lot of discussion in the literature as to whether traces of NP-movement are subject to Condition A or not. There is some evidence to suggest that Condition A cannot capture the very strict locality requirements on these traces, however, and that the ECP is required to do the job. On this view, NP-traces would not be subject to Condition A at all. For a detailed discussion of this problem, see Lasnik (1985) and Chomsky (1986).

not yet moved the arguments from their θ-positions and therefore an optimally simple θ-Criterion can be postulated:

(24) θ-Criterion (D-Structure)
Each argument is in a θ-position and
each θ-position contains an argument

Here is the problem. If the θ-Criterion holds at D-Structure, then the heads of A'-chains (i.e. elements like *wh*-phrases and quantifier/operator phrases) will have to count as arguments for it, since these appear in θ-positions at this level. Given standard assumptions, however, these elements are *not* arguments at LF: the θ-role assigned to them at D-Structure is taken up by the associated variable. Let me give an example to make this clear. Example (25i) has D-structure (25ii) and LF-representation (25iii):

(25) (i) who did John see
(ii) $[_{CP}\ [_{IP}$ John <past> $[_{VP}$ see who]]]
(iii) $[_{CP}$ who$_i$ do+<past>$_j$ $[_{IP}$ John t_j $[_{VP}$ t_i' $[_{VP}$ see t_i]]]]

By the θ-Criterion *who* in (25ii) will have to be an argument, as it occupies a (D-Structure) θ-position. At LF, however, it is t_i that counts as an argument, i.e as a referential category or *R-expression*[16]. The theta-role is not "inherited" by the *wh*-phrase at the head of the chain. There are several reasons for this assumption, but we will not go into them here.

Over the past couple of years several linguists have argued that the θ-Criterion does not just hold at D-Structure, but that it holds at LF as well. Indeed, according to some, it only needs to be stated at LF. The principle would have the following form at LF:

(26) θ-Criterion (LF)
Each θ-position is (chain-)related to a unique argument and
each argument is (chain-)related to a unique θ-position

This version of the θ-Criterion is incompatible with a "strong" version of the derivational view of move α, as will become clear shortly.

Brody argues (i) that the θ-Criterion holds *only* at D-Structure, (ii) that it refers *only* to D-Structure arguments, and (iii) that (26) is incorrect. Thus, while the notion of LF argument (i.e. R-expression) may be useful, it is not one that the θ-Criterion is concerned with.

The D-Structure θ-Criterion ensures a one-to-one mapping between θ-positions and arguments simply by virtue of the topology of syntactic trees: one

[16]The set of R-expression include noun phrases that are in some intuitive sense potentially referential (e.g., *John*, *the cat*, etc.) and variables, where we define variable as follows: α is a variable iff α is an operator-bound trace.

position can contain only one argument, one argument can only be in one position. At LF, however, where θ-positions can be related to arguments through chains, uniqueness does not follow, and if one drops the LF θ-Criterion, one would expect to see cases where a chain contains more than one LF-argument (R-expression). Brody argues that such cases do in fact exist. As he points out, "the analysis of the English adjectival complement constructions of the *easy-to-please* type is another longstanding problem of GB-theory". In this construction the matrix subject is non-thematic as shown by (27i), but an argument can appear in it as in (27ii):

(27) (i) it's easy [to please John]
(ii) John is easy [to please t]

The cooccurrence of an argument in the matrix subject position and of a gap in the complement clause is non-accidental:

(28) (i) * John is easy [to please Mary]
(ii) ok It's easy [to please t] (only with *it* as non-dummy)

In other words, the argument appears in the matrix subject position if and only if the complement contains a related gap. Thus, the structure has all the properties of movement. The fact that both the position of the gap and that of the moved element are A-positions makes the movement similar to NP-movement. On the other hand, the gap is in the object position of a non-passivized transitive verb, the movement does not obey the ECP (that is, assuming improper movement is disallowed), and there is an intervening subject: all properties not normally found with NP-movement. Chomsky (1977) pointed out that the construction exhibits all major features of *wh*-movement and proposed an analysis of these facts that involves empty operator movement:

(29) John is easy [Op$_i$ [to please t$_i$]]

However, since Chomsky (1981), when the θ-Criterion and the assumption that variables are arguments were introduced, the construction has been a real problem. This is because both the matrix subject and the variable t_i are arguments in violation of the θ-Criterion, if this is construed as entailing a one-to-one mapping between arguments and θ-roles at LF. On Brody's view of the matter, this is not a problem at all. It is in fact exactly what one expects to find, given that the θ-Criterion holds at D-Structure only.

According to Brody the real problem with the *easy-to-please* construction is created by the ban on improper movement: how is the matrix subject in (29) related to the operator and the variable? As the subject occupies a non-θ-position, it must have moved there. But the only possible D-Structure position for the subject is the gap in the complement clause which appears to be the launching site for the empty operator. As Brody points out:

> "We have a rather curious and otherwise unattested peculiarity: an argument materializing in a non-theta A-position with no proper D-structure source. The argument must also somehow be connected to the operator-variable chain if for no other reason than to ensure correct interpretation. If we take this to be accomplished by some construal rule, then we also have an otherwise unattested construal rule relating an argument in a non-theta position to a thematic position.
>
> But of course we have a rule with precisely these properties, namely movement. So the problem appears to be located in the assumption that improper movement is generally prohibited."
> (p.4)

So Brody's next step is to drop the general ban on improper movement and to derive (29) through movement, as in (30), where (30i) is the D-structure and (30ii) is the derived S-structure:

(30) (i) $[_{IP}$ NP is easy $[_{CP}$ $[_{IP}$ PRO to please John]]]
(ii) $[_{IP}$ John$_i$ is easy $[_{CP}$ t_i $[_{IP}$ PRO to please t_i]]]

Of course, it must be ensured that improper movement derivations that need to be excluded are ruled out. Consider again (21), which I repeat here as (31) for convenience:

(31) * $[_{IP}$ John$_i$ $[_{VP}$ t_i $[_{VP}$ seems $[_{CP}$ t_i $[_{IP}$ Mary $[_{VP}$ t_i $[_{VP}$ saw t_i]]]]]]]

It is essential for what follows that I first point out that for Brody any intermediate trace in an A'-position can be an operator. Whether such a trace is in fact an operator or not depends on whether the head that governs the trace licenses an operator (i.e. whether the head selects an operator as a lexical property). This idea may seem somewhat strange at first, but the examples below show the lexically governed nature of this licensing:

(32) (i) ok John is easy [Op [to please]]
(ii) * John is feasible [Op [to please]]
(iii) ok John is impossible [Op [to please]]
(iv) * John is not possible [Op [to please]]

Crucially, the trace governed by *seem* in (31) is not licensed as an operator. Given this, (31) could be straightforwardly ruled out by Condition C of the Binding Theory, if the trace following *saw* is taken to be an R-expression. Because if it is an R-expression, then it has to be A-free, which it is not (it is A-bound by *John*). On standard assumptions, the trace is not an R-expression, however. For a trace to be an R-expression, it has to be operator-bound. And we have just seen that none of the intermediate traces in (31) qualifies as an operator. Similar problems arise with ungrammatical cases of improper movement involving chains that terminate in a caseless position. Consider (33) (irrelevant details omitted):

(33) * [$_{IP}$ John$_i$ [$_{VP}$ seems [$_{IP}$ it appears [$_{CP}$ t_i' [$_{IP}$ t_i to [$_{VP}$ like Mary]]]]]]

Here t_i is the caseless non-operator-bound subject of an infinitive and hence not an R-expression by anyone's book. Therefore, Condition C will not rule (33) out either.

In order to account for these cases, Brody proposes the Chain Uniformity Condition:

(34) Chain Uniformity Condition
The (sub-)chain $(\alpha_1,...,\alpha_i)$ of $(\alpha_1,...,\alpha_i,...,\alpha_n)$ must be uniform unless $(\alpha_i,...,\alpha_n)$ contains an R-expression

For clarity's sake let us also define Uniformity and R-expression:

(35) Uniformity
A (sub-)chain is uniform iff it involves only A-positions or only A'-positions

(36) R-expression
A category α is an R-expression iff
(i) α has referential features, or
(ii) α is an operator-bound trace

The uniformity condition (34) rules out (31) and (33) as required. In each case the foot of the chain is a trace in A-position that is not operator-bound. Hence, the chain of which they are part must be uniform and neither of these chains is. Structure (30ii), by contrast, is fine. The foot of the chain is operator-bound by the trace governed by *easy* and hence the non-uniformity of the chain is licensed.

In the following subsection I will show how Brody's approach to movement can help us constrain the mapping from D-Structure to S-Structure.

2.2.5 Using the Chain Uniformity Condition to constrain move α Returning now to our main theme, recall that we were looking for ways to constrain the *move_α** relation in such a way that we could avoid the generation of chains with improper movement. This was part of a more ambitious goal: constraining the *move_α** relation to the point where it generates all and only well-formed outputs without backtracking.

Our excursion in the previous section has far-reaching consequences. Some of these are of importance for the overall approach to the GB parsing problem that I am taking and will be discussed later. There are at least three results that are of direct importance for our present concerns. First, Brody's theory, if essentially correct, tells us that improper movement is not something to be avoided, but instead has instances that produce a well-formed result. Second, it makes it unnecessary to try to bring Condition C of the Binding Theory to bear on the application of move α. In fact, under Brody's theory this would not even give us the result that we want, because Condition C does not rule out all unacceptable

cases of improper movement. Third, the Chain Uniformity Condition is a fairly straightforward recipe for how we should reformulate the *move_α** relation to suit our purposes. When we are trying to prove that some S-structure can be derived from some D-structure through the application of move α, then we are in fact trying to prove the existence of a series of tree structures (of length greater than zero) such that (i) the first tree is a D-structure, (ii) each following tree is related to the previous one by one application of move α, and (iii) the final tree is a valid S-structure. What this means in terms of chain uniformity is this: either (i) one tree is related to the next by a uniform movement (i.e. from A-position to A-position or from A'-position to A'-position) or (ii) one tree is related to the next by non-uniform movement, in which case we must prove in addition that there exists an R-expression in the sub-chain already derived (i.e. a c-commanding operator binding a trace). I give a translation into first-order logic below:

(37) $(\forall T_0,T_1,I)\ move\_\alpha^*(T_0,T_1,I) \leftrightarrow (T_0 = T_1 \vee$
$[[(\exists T_2)\ uniform\_move\_\alpha(T_0,T_2,I) \wedge move\_\alpha^*(T_0,T_1,I))] \vee$
$[(\exists T_2)\ non\_uniform\_move\_\alpha(T_0,T_2,I) \wedge$
$move\_\alpha^*(T_2,T_1,I)) \wedge$
$(\exists Op,S)\ (operator(Op,I,T_2) \wedge$
$trace(S,I,T_2) \wedge$
$c\_command(Op,S,I,T_2))]]$

I have provided the variable *i* to indicate that the *move_α** relation must be satisfied for all indices and because the predicates *operator*, *trace* and *c-command* must have an index as their argument. What (25) says is that two trees are either related through uniform movement or through non-uniform movement. In the latter case, the result is only valid if the resulting tree contains a c-commanding operator binding a trace. Since this requirement is checked immediately after an application of improper movement, compatiblity with the CUP is guaranteed.

Compiling the Chain Uniformity Principle (CUP) into the *move_α** relation leads to a considerable reduction in the search space for this predicate. It is difficult to estimate how big the reduction is in the general case, however, because this is dependent on the number of landing sites that are A-positions. In the worst case the number of A-positions and A'-positions that are potential landing sites would be roughly equal. Assuming the earlier modification, which restricts movement of a category with bar level *b* to a landing site with bar level *b*, the number of possible derivations for *any* node in a tree of size *n* (*n* the number of nodes) without compilation of the CUP would be roughly

$$\sum_{k=0}^{1/3n} \frac{1/3n!}{(1/3n-k)!}$$

With the CUP compiled into the *move_α** relation, this would be reduced to

$$\sum_{k=0}^{1/6n} \frac{1/6n!}{(1/6n-k)!}$$

This is a substantial result. One might object that the gain in efficiency will be partly or even completely lost as a result of the amount of tree walking that will have to be performed by the revised version of *move*_α*. We return to this matter directly below, where we discuss similar objections in connection with compiling Subjacency into *move*_α*.

2.2.6 Using Subjacency to constrain move α I already mentioned in passing that it should certainly be possible to compile Subjacency into the *move*_α* relation. Whereas the CUP was concerned with specifying potential landing sites irrespective of locality consideration, compiling Subjacency into *move*_α* will have the effect of ruling out many potential landing sites that do not satisfy certain locality requirements. I will not attempt to give a FOL version of this version of *move*_α* here, but instead will sketch its form and discuss an objection that might be raised against this approach.

A key part of the definition the *move*_α* predicate that tests nodes to see if they are possible landing sites with respect to Subjacency will have to be some sort of tree-walking predicate. Tree-walking will be frequent, since for every application of move α we have to find a suitable landing site. Exactly the same objection might be raised with respect to the CUP-compilation proposal. One might argue that this could make the modified *move*_α* relation quite inefficient. Several remarks seem appropriate here. First, tree-walking will be frequent as well if we have to test at S-Structure whether all nodes satisfy Subjacency (similarly for the CUP at LF). Although in this case we have to walk the tree only once for every S-structure, the tree-walk will have to be done over and over again if the old version of *move*_α* produces one invalid S-structure after the other (as we have reason to believe it will). Second, we might very well be able to come up with a clever tree-walk for our revised *move*_α* that does not consider each and every node in the tree. Starting from the trace that constitutes the launching site, the tree-walk could look for the *nearest c-commanding* landing site meeting the locality requirement imposed by Subjacency. In fact, this is precisely why we want to compile Subjacency into *move*_α* in the first place. Third, if we let such a "nearest node" search precede testing for satisfaction of the CUP, we would also further reduce the search space associated with this principle. On top of that, it seems quite likely that we could fold the CUP and the Subjacency components of the *move*_α* predicate so that the tree-walk is performed only once for every application of move α. Finally, tree-walking will be further reduced if we take advantage of the standard co-routining techniques to rule out structures that violate LF constraints at an early point in the proof.

2.3 Summary and conclusions

I have sketched a rather unconventional generate-and-test procedure that is provided with the input string at D-Structure to bring its guesswork down to reasonable proportions. Better than that, this generate-and-test method does not suffer from the decidability problems that make the naive generate-and-test method virtually useless. Building the D-structure first has the advantage that the θ-Criterion and X'-theory can be brought to bear on the parsing problem at an early stage.

I also pointed out that the "derivational" view of the mapping between levels in GB-theory suggests novel ideas about grammar compilation. The compilation methods I have suggested are not aimed at compiling away one or more levels of representation, but instead at compiling away the vast search space introduced by the completely unrestricted rule move α. The key idea I worked with was that some principles of grammar can be regarded as exclusively "chain-related". From the point of view of the grammar, this means that these constraints, which are generally assumed to be constraints on representations, can be reinterpreted as constraints on derivations. From the point of view of parsing, such principles are natural candidates for compilation into move α. Some effort was put into showing how such compilation steps might be realized.

References

Abney, S. (1986). Licensing and Parsing. *Proceedings of NELS 17*, University of Massachusetts at Amherst, Amherst MA.

Berwick, R. (1987). Principle-Based Parsing. Technical Report 972, MIT Artificial Intelligence Laboratory.

Berwick, R. and A. Weinberg (1984). *The Grammatical Basis of Linguistic Performance*. Cambridge, MA: MIT Press.

Berwick R. and A. Weinberg (1985). Deterministic Parsing: A Modern View. NELS 18, pp.15-33.

Brody, M. (1990). Case Theory and Argumenthood. Ms. University College London. [to appear in *Linguistic Inquiry*].

Chomsky, N. (1977). On Wh-Movement. In A. Akmajian, P. Culicover and T. Wasow, eds., *Formal Syntax*. New York: Academic Press.

Chomsky, N. (1981). *Lectures on Government and Binding*. Dordrecht: Foris.

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

Johnson, M. (1989). Parsing as Deduction: The Use of Knowledge of Language. *The Journal of Psycholinguistic Research* 18.1., pp.105-128.

Kashket, M. (1987). *A Government-Binding Based Parser for Warlpiri, a Free-Word Order Language*. Technical Report, MIT Artificial Intelligence Laboratory.

Kolb, H.-P. and G. Thiersch (1990). Levels and Empty Categories in a Principles and Parameters Approach to Parsing. ITK Research Report No. 19, Tilburg University.

Koot, J. van de (1990). *An Essay on Grammar-Parser Relations*. Dordrecht: Foris.

Lasnik, H. (1985). Illicit NP Movement: Locality Conditions on Chains?. *Linguistic Inquiry* 16, pp.481-490.

Mackworth, A.K. (1977). Consistency in Networks of Relations. *Artificial Intelligence 8*: pp.99-118.

Mackworth, A.K. (1987). Constraint Satisfaction. In S.C. Shapiro, ed., *Encyclopedia of Artificial Intelligence*. New York: Wiley.

Marr, D. (1982). *Vision*. San Francisco, CA: Freeman.

Pereira, F. and D. Warren. (1983). Parsing as Deduction. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*. MA: MIT Press.

Ristad, E. (1991). *A Constructive Complexity Thesis for Human Language*. Technical report, Princeton University. [revised version of PhD dissertation, MIT (MIT AI Lab, Technical report 1260)].

Stabler, E.P., Jr. (1990a). *The Logical Approach to Syntax*. MIT Press, forthcoming.

Stabler, E.P., Jr. (1990b). Relaxation Techniques for Principle-Based Parsing. Talk held GB-Parsing Conference, Geneva.