Strong Generative Capacity, Weak Generative Capacity, and Modern Linguistic Theories

Robert C. Berwick

MIT Artificial Intelligence Laboratory
Cambridge, MA 02139

Introduction

What makes a language a *natural* language? A long-standing tradition in generative grammar holds that a language is natural just in case it is *learnable* under a constellation of auxiliary assumptions about input evidence available to children. Yet another approach seeks some key mathematical property that distinguishes the natural languages from all possible symbol-systems. With some exceptions – for example, Chomsky's demonstration that a complete characterization of our grammatical knowledge lies beyond the power of finite state languages – the mathematical approach has not provided clear-cut results. For example, for a variety of reasons we cannot say that the predicate *is context-free* characterizes all and only the natural languages.

Still another use of mathematical analysis in linguistics has been to diagnose a proposed grammatical formalism as too powerful (allowing too many grammars or languages) rather than as too weak. Such a diagnosis was supposed by some to follow from Peters and Ritchie's demonstration that the theory of transformational grammar as described in Chomsky's *Aspects of the Theory of Syntax* could specify grammars to generate any recursively enumerable set. For some this demonstration marked a watershed in the formal analysis transformational grammar. One general reaction (not prompted by the Peters and Ritchie result alone) was to turn to other theories of grammar designed to explicitly avoid the problems of a theory that could specify an arbitrary Turing machine computation. The proposals for generalized phrase structure grammar (GPSG) and lexical-functional grammar (LFG) have explicitly emphasized this point. GPSG aims for grammars that generate context-free languages (though there is some recent wavering on this point; see Pullum 1984); LFG, for languages that are at worst context-sensitive. Whatever the merits of the arguments for this restriction in terms of weak generative capacity – and they are far from obvious, as discussed at length in Berwick and Weinberg (1983) – one point remains: the switch was prompted by criticism of the nearly two-decades old *Aspects* theory.

Much has changed in transformational grammar in twenty years. Modern transformational grammars no longer contain swarms of individual rules such as Passive, Raising, or Dative. The modern government-binding (GB) theory does not reconstruct a "deep structure", does not contain powerful deletion rules, and has introduced a whole host of new constraints. Given these sweeping changes, it would seem appropriate, then, to re-examine the Peters and Ritchie result, and compare the power of the newer GB-style theories to these other current linguistic theories. That is the aim of this paper. The basic points to be made are these:

- Since modern transformational grammars do not contain the powerful deletion rules available in the *Aspects* theory and need not explicitly reconstruct an underlying deep structure, they are not immediately subject to the Peters and Ritchie results. Thus the fears recently advanced by Bresnan and Kaplan (1982: xli–xlii) or Johnson-Laird (1983: 280) simply do not hold.
- Because modern transformational grammars use **traces** to mark the site of displaced constituents, the size of underlying structures that need be recovered for language recognition are just linearly larger than their corresponding surface sentences. Indeed, it appears that deep structures ("D-structures" in the current theory) need not be built at all to test grammaticality.
- Modern transformational grammars seem more restricted than theories like LFG, not less restricted, in the sense that the agreement predicates available in a modern transformational theory are defined solely over **unordered sets** of features, rather than, as in the lexical-functional theory, over **hierarchical trees.** Agreement ("unification") over trees adds extra power to the

0362-613X/84/030189-14$03.00

lexical-functional formalism. The result is that there are some strikingly unnatural grammars that lexical-functional grammars can describe, but not GB grammars. This result about strong generative capacity shows up on the weak generative capacity side: GB grammars cannot generate some strictly context-sensitive languages that can be easily generated by lexical-functional grammars.

This paper is organized as follows. Section 1 reviews some of the basic formal and linguistic examples demonstrating that the excess power of the *Aspects* theory comes from unbounded deletion. It then shows why this power is not permitted in current transformational theories. Section 2 turns to a general analysis of the power of government-binding grammars. Section 3 compares the strong and weak generative capacity of lexical-functional grammar and transformational grammars. It aims to pinpoint just why lexical-functional grammars are more powerful than government-binding grammars. Section 4 concludes with some speculations about the precise formal characterization of natural languages. More generally, these results suggest a different role for the formal analysis of natural languages. Instead of trying to fit natural languages into some pre-defined mathematical or formal mold, this revised strategy aims to discover the properties of natural languages first, and then characterize them formally. The results here may be regarded as the first fruits of this strategy, applied to current linguistic theories.

1. Unbounded Deletion, Past and Present

It has long been recognized that the possibility of unbounded deletion is at the root of the computational power of *Aspects* style transformational theories. If what a machine must do to recognize whether or not a given sentence (surface string) is in the language generated by some transformational grammar is to recover its deep structure, and if deep structures can be arbitrarily large compared to the surface strings derived from them, then the recognition procedures for such languages are not even recursive.

Before describing Peters and Ritchie's formal characterization of this connection between deep structure length and the complexity of recognition, it would be valuable to give some insight into just why this connection should hold. Recall that a **recursive set** is one where membership in the set can be determined in some finite (though perhaps large) amount of time. Here, the set we have in mind is the language generated by some transformational grammar, *L(TG)*; given some sentence *s*, our job is to calculate $s \in L(TG)$ and return a *yes* or *no* answer in some finite amount of time. Also recall that a set is **recursively enumerable** (r.e.) if, whenever *s* is in fact in *L(TG)*, there is a procedure such that the answer *yes* can come back in some finite amount of time, but if *s* is not in the language, we have no such guarantee; the procedure could just run forever.

The key insight connecting length of deep structure to recursive enumerability comes from examining the conditions under which a computation could run forever. If we use a standard Turing machine model of computation, one thing that could happen is that the machine could just keep using more and more new tape cells, moving a step each time. This could go on forever. So one way to obtain unbounded computation time is to use unbounded space. If we substitute for the "tape cells" of the Turing machine the number of embedded s or np cycles in some arbitrarily large deep structure, and *if we must recover this deep structure in order to figure out whether or not the sentence is in the grammar,* then we have our correspondence between unbounded deep structures and unbounded time for computation.

But is this the only way to achieve unboundedly long computations? Why not just have the machine shuttle back forth along some fixed sequence of tape cells, using the same space but looping forever? This is certainly possible, but in this case one can show that the number of distinct machine configurations is bounded above by the cross-product of a fixed number of possible moves times a fixed number of possible cell contents. But this means we could "shut off" the machine after this number of time steps (counting each Turing machine move as a tick of the clock), since the machine cannot do anything new after this number of moves.[1] In other words, given an upper bound on the space a machine uses, we can fix an upper bound on the length of time the machine can ever use without looping forever.[2]

In short then, the only way to get non-recursive computations is by using unbounded space. In the transformational analog, Peters and Ritchie (1973) connected recognition complexity to the possible difference in length between deep structures and surface strings:

> Let *G* be a transformational grammar. Let f_G be the cycling function of *G*, where $f_G x$ is *0* if *x* is not in *L(G)*, and otherwise is the least number *s* such that *G* assigns *x* a deep structure with *s* subsentences. If f_G is bounded by an elementary (primitive) recursive function, then *L(G)* is elementary (primitive) recursive. (In fact, if f_G is linear, then *L(G)* is in a still smaller class.) If the cycling function is not bounded, then *L(G)* is not even recursive.

It is the possibility of arbitrary deletion that makes a surface sentence arbitrarily "shorter" than its corresponding underlying deep structure. Lapointe (1977), in an excellent review, sums up the situation:

> Putnam noted that early theories of transformations allowed grammars which could generate any r.e. language (whether recursive or not). The chief reason for this was that early theories allowed arbitrary deletions and substitutions in the course of a derivation. Arbitrary permutations or copying could never cause a grammar to generate a nonrecursive set, for if τ_i and τ_{i+1} are successive steps in a derivation such that τ_{i+1} arises through the application

[1] The "clock" takes up a bit of extra space – log space – since it has to count!

[2] This standard result may be found in Hopcroft and Ullman (1979: 300-301).

> of a permutation or copying rule [from] τ_i then . . . the number of terminal symbols in τ_{i+1} will be at least as great as [the number of] terminal symbols in τ_i. But this property, that successive steps in a derivation do not "shrink" in length, is the basic defining characteristic of context-sensitive grammars. Therefore only the application of rules which reduce length (that is, deletions and substitutions) could cause a grammar to generate a non-CS [context-sensitive rcb], and perhaps a nonrecursive, language. (1977: 228)

We can exhibit this result in a compact form. It is well known that any r.e. set can be described as the homomorphic image of the intersection of two context-free languages (Ginsburg, Greibach, and Harrison 1967). That is,

$$L0 = H(CFL_1 \cap CFL_2)$$

Recall that a homomorphism is simply a "respelling" of the symbols of a language. The key point is that the homomorphism required here permits the deletion of unbounded strings of symbols, that is,

$$H(z) = \lambda$$

where λ is the empty string. In fact, all the proofs demonstrating the power of *Aspects*-style transformational grammars make use of this erasing power in one fashion or another. The remainder of this section reviews two of these demonstrations in the literature, one by Peters (1973), and one by Kimball (1967). (Another demonstration that *Aspects*-style TGs can generate any r.e. language, given by Salomma (1971), also uses unbounded deletion, but is similar to Kimball's approach and will not be discussed here.) The point of going through the examples in detail is to show exactly how each proof relies on unbounded deletion, and why it is that each does not go through under the assumptions of the current government-binding theory. The basic reason for the change is that unlimited erasing or deletion is no longer allowed. Indeed, as the next section will make clear, only a *linear* amount of erasing is permitted in current theories. This insight is the key to the analysis of the modern theory.

We begin with Peters's 1973 demonstration. Peters gives a specific example showing just how "large" deep structures can be associated with "short" surface sentences. Again, copying and deletion are the culprits. Peters's example relies on the "Equi np deletion" analysis of sentences such as these:

1. Their sitting down promises to steady the canoe.

On this account, such sentences have an underlying structure that explicitly reconstructs the missing np subject of the embedded complement *to promise*:

2. $[_{NP}$ Their sitting down] promises $[_S$ $[_{NP}$ their sitting down] to steady the canoe].

Note that this sentence consists of *three* S phrases: the root S and two embedded S phrases (the subject NP of the matrix clause and the subject NP of the complement of the VP). The subject NP of the VP complement is deleted under structural identity with the matrix subject NP. This deletion follows the "recoverability of deletion" constraint. Peters next builds a surface string that has a large associated deep structure by embedding this sentence recursively in a construction of the same type, that is, a sentence that has a matrix subject NP structurally identical to the subject NP of a verb complement. But the subject NP has more than two S phrases (three). Given identity between subject NP of the matrix and the subject of the complement, it follows that at the level of deep structure the subject NP of the complement must have the same number of subsentences as the subject NP of the matrix, here, three. The new sentence given below must have a deep structure with more than $2^2 = 4$ S phrases in all:

3. Their sitting down promising to steady the canoe threatens to spoil the joke.

Clearly, as Peters notes, we can carry out this embedding over and over again. Each time the number of deep structure subsentences is at least doubled, because of the assumption that the complement NP subject is identical to that of the matrix subject. If we let *ds(n)* be the size of the deep structure corresponding to such a sentence of length *n*, then we have the inductive formula that $ds(n) > 2ds(n-1)$. If we solve this formula, we find that the number of deep structure subsentences grows as an exponential function when compared to the length of the surface string, exactly the sort of sentence that was to be constructed. If the sentence recognizer must reconstruct this entire deep structure in order to determine language membership, then at least this much space, and hence at least this much time, will be required, just to write down the deep structure.

Interestingly, the argument does not work under current versions of transformational theory. The simple reason is that we no longer explicitly copy material to reconstruct a deep structure; in fact, we no longer rebuild deep structure at all. In place of the literally duplicated subject complement NPs, we have an empty category placeholder, PRO, indexed to the proper antecedent NP as appropriate.[3]

4. $[_{NP}$ $[_{NP}$ Their sitting down], promising [Pro_i to steady the canoe]]$_j$ threatens [Pro_j to spoil the joke]

Crucially, the indexed Pros are not "nested". What does this mean and why does this matter? Pro_j is indexed to the *entire* matrix subject NP *their sitting down promising to steady the canoe.* But it *does not* contain as a subpart the PRO corresponding to *their sitting down* (although it may be indexed to a subpart of a long antecedent string). The underlying predicate-argument structure is fixed without building up an explicit representation of antecedents in the embedded clause, what used to be called "deep structure."

[3] The other possibility is that the empty category is a **trace,** the result of the movement of an NP from an argument position like the direct object of a transitive verb. Here the empty category is PRO rather than **trace** because the subject NP position in the complement is not *governed* by tense or the matrix verb, but the reader may safely ignore this detail for our purposes here.

By changing what the representation looks like we have avoided the problem of exponential space growth. At each step we add just a *single* new element to the reconstructed structure, the new PRO. Our new inductive equation is simply $ds(n)=ds(n-1)+1$ – a linear increase in the size of reconstructed forms, compared to the surface sentence lengths. In fact, if we ignored the bracketing and just counted the PROs, at each step we also add a new word (the new verb) and the underlying representations are always just a fixed **constant** larger than the corresponding surface sentences.

One subtle point still remains. At each step we add an *indexed* element, PRO_i. The index itself must grow as the number of PROs increases. If we assume a standard binary encoding, an index of size i will take $\log_2 i$ space to write down, not the constant space implicitly assumed just above. Since $\log i < i$, at worst the space added for each embedding will be proportional to i. Summed over n possible embeddings, this is at worst n^2 space, not exponential space. In the next section we shall see how this representation of so-called "empty categories" works in general.[4]

Peters's example centered on a "natural" example that exhibited exponential deep structure growth. We next turn to a more artificial example, but one showing how arbitrarily large deep structures may be used to generate any r.e. language. Kimball (1967) does this by exhibiting a transformational grammar that meets a variant of the Ginsburg, Greibach, and Harrison theorem (due to Haines, cited in Kimball 1967: 185). In brief, Kimball sets up a base context-free grammar to generate two trees, rooted at S_1 and S_2, corresponding to the two context-free languages demanded by the homomorphism theorem (CFL_1 and CFL_2) and a third tree dominating these two that eventually "simulates" the homomorphism H. S_1 dominates a terminal string labeled x and S_2, a terminal string labeled y. Dominating these two subtrees is a third tree that, besides x and y, dominates a terminal string z. The idea is to use a single transformation to successively check that the first member of x matches the first member of y and z; if so, this element is erased in x and y. If all elements match, and we are at the end of z (indicated by a special symbol), then the two strings are identical; this step carries out the intersection of the two context-free languages. A final transformation performs the required homomorphism. Figure 1 depicts the overall scheme. It is important to point out that both S_1 and S_2 generate context-free languages that are self-embedding, of the general form $a^i c a^i$. Thus they must exhibit recursion on some nonterminal node in the relevant context-free grammar.

As Kimball notes, the strings x and y are deleted under identity with z, so nothing is amiss here in the *Aspects* theory. X and y are arbitrarily long. The underlying "deep structure" (the context-free base) is arbitrarily larger than the resulting surface string, namely, some part of z that remains after the homomorphism does its work.

What happens to this example in a modern transformational theory? The key point is that the modern theory does not have a deletion operation like the one just presented. Instead, a constituent is *moved* from one position to a "landing site" within its own cyclic domain or to the next higher cyclic domain.[5] In English, the cyclic nodes are S and NP.[6] When a node is moved, it leaves behind a **trace**, denoted e, of the same category as the displaced constituent, but with no phonological features. (Thus the trace is not "pronounced" and does not show up in the surface sentence.) The trace is co-indexed to the displaced constituent, as indicated by a subscript. For example, given the sentence,

5. John bought what

we could move *what*, yielding (after some adjustment with the auxiliary verb),

6. What did John buy e_i

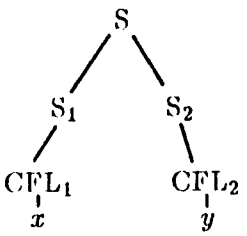
Now consider Kimball's tree structures again. Since S_1 and S_2 are true recursive sub-trees, in a trace-oriented theory the way that we would get deletion would be to successively move elements of x and y to higher and higher phrases, leaving behind traces (denoted by e_i) as we go. Schematically, our output structure would have to look something like that in figure 2, where R indicates a cyclic node. As it stands though, this structure is impossible because it requires traces to be linked to elements that are "too far away": according to a key constraint of the modern transformational theory, the subjacency constraint, the linking can cross at most *one* cyclic node. Since all recursion must eventually pass through S or NP nodes, subjacency must be violated by the trees pictured in figure 2. Put another way, the rule that "erases", for example, x_1, now must move x_1 across many S or NP nodes, and this movement is not directly possible. An alternative is to move x_i **successive cyclically**, up the chain of R nodes one step at a time. But there are only two ways to do this: either we wind up moving more and more nodes at each step – at the n^{th} step we move n nodes, which must "land" at n spots at the next higher cyclic domain – or we collapse how many nodes we move by adjoining some of the moved elements together. Figure 3 shows both possibilities.

Both solutions are ruled out in current theories of transformational grammar. The movement of an arbitrary number of nodes in a single cycle is impossible because it calls for an arbitrary number of "landing sites" in domain $n+1$. In fact, there can only be a finite number of such possibilities, as specified by a set of context-free base rules. For example, we can move an NP to a subject or object

[4] There is another solution to the index growth problem, one that will be required later on. Suppose that each indexed NP or PRO is in effect a distinct element of the grammar's vocabulary. Thus the grammar allows a denumerable infinity of "pre-indexed" elements NP_1, NP_2, ... This is not such a strange proposal, since the index is not used for any syntactic process, but simply for co-indexing. As we shall see, this same proposal is made, usually implicitly, in most current theories, for example, in the lexical-functional theory.

[5] In the next section we shall take a slightly different position and define admissible annotated surface structures without literal movement.

[6] For our purposes here, cyclic nodes are those that exhibit recursion.



Transformation:

structural condition (roughly):
$[_{S_1} Xx]$ $[_{S_2} Yy]$ Zz
$x = y = z$; x, y, z are terminal symbols

structural change:
delete x and y

Figure 1. Kimball's transformation to generate any r.e. set.

position, a *wh* phrase to a comp position (the position occupied by *that* in *I know that Mary likes ice cream*).[7] But we cannot move n nodes in a single cycle, because there will not be enough places to put the moved constituents.

The second solution is also ruled out. Either the adjoined NPs linked to the traces violate subjacency (as pictured), or else we must also adjoin at each step i a copy of the $i-1^{st}$ trace to the i^{th} trace. But this last method is also barred, because we do not admit "nested" traces, or tree structures that dominate some arbitrarily deep sequence of nested empty elements. In other words, a trace can be co-indexed at its "top-level" to a displaced constituent, but is otherwise "opaque"; it has no interior structure. There are in fact good linguistic reasons for a principle banning nested traces (see Hornstein 1984). Section 3 probes the formal implications of this constraint in more detail.

It is hard then to see how the required trace structure linking x and z could even be built. But this is just the first step in Kimball's proof. Enforcing equality between x and y looks even harder. Of course, this by no means shows that there is *no* way to carry out Kimball's construction, but it does hint at some of the difficulties in a revised grammatical framework that does not permit the same liberties with deletions as *Aspects*, and does not rely on an explicitly reconstructed D-structure.

2. The Complexity of Modern Transformational Grammar

As we have seen, the crux of the problem with *Aspects*-style transformational grammars is deletion, and, more pointedly, the demand to recover unboundedly large deep structures in order to determine sentence-hood. The proofs of intractability all hinge on the assumption that the job of the parser is to recover a literal copy of deleted elements. If this assumption is not needed, then the job of the recognizer could well be easier. The modern theory requires only the recovery of a trace– or PRO–augmented structure, an "annotated surface structure". This makes a difference. As Lapointe (1977) shows, it makes the recognition problem for such languages recursive. Whatever the merits of their arguments on other grounds, Lapointe's result renders moot Bresnan and Kaplan's concerns (1982: xli) about the non-recursiveness of transformational theory, since their criticisms apply only to the older *Aspects* theory. This is our first conclusion.

Much more than this can be said. If the recognizer does not have to recover full deep structures, then its job could be much easier, as observed by Peters and Ritchie 1973:

> Putnam proposed that the class of transformational grammars be defined so that they satisfy a "cut-elimination" theorem. We can interpret this rather broadly to mean that for for every grammar G_1 in a class there exists $_2$ such that (i) $L(G_1) = L(G_2)$ and (ii) there is a constant k with the property that for every x in $L(G_2)$, there is a deep phrase marker ϕ underlying x with respect to G_2 such that $l[d(\phi)] < kx$.
> (1973: 81–82)

Here, the notation $l(x)$ stands for "length of", while $d(\phi)$ is the "debracketization" of the deep structure. The debracketization consists of terminal elements sans right and left brackets, but with traces and PROs. As Peters and Ritchie go on to say:

> We now see that any grammar satisfying such a cut-elimination theorem generates a language which more than being recursive is context sensitive. This is so because a nondeterministic linear bounded automaton can determine both that a labeled bracketing ϕ is strongly generated by a context sensitive grammar and that it underlies a given string x if the automaton has enough tape to write ϕ.
> (1973:82)

How would such a linearly-bounded recognizer work? Roughly, it would use a kind of "analysis by synthesis": given a sentence of length n, it would mark out a length of input tape kn, k a constant depending on the transformational grammar. The machine would be guaranteed that annotated surface structures could not get larger than this.

[7] See the next section for more on "landing sites."

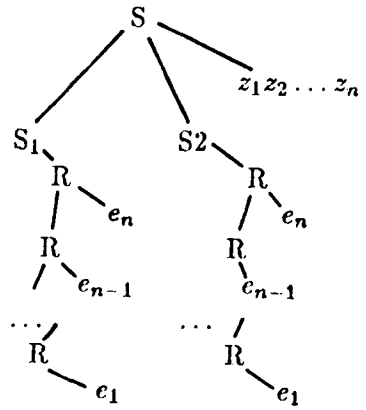


Figure 2. Traces and the Kimball construction.

The machine would then use its nondeterministic power to "guess" all possible annotated surface structures less than or equal to this length, now with the proviso that one of them must be a correct underlying structure if the sentence in question is in fact in the language generated by the grammar. Since the number of (NP or S) cycles in each structure is bounded, we may simply try all possible transformational rules (again nondeterministically) to produce possible surface sentences, one at a time. If there is a match, then the sentence is in the language; if all structures less than our bound are tested an fail, then the sentence is not in the language.[8]

What then of modern transformational grammar? We claim that D-structure need not be reconstructed at all to determine grammaticality. This may be a surprise for some readers accustomed to the older picture of a transformational grammar, where annotated surface structure is just the result of mapping from D-structure under the operation of Move α. But it is nonetheless true. Chomsky (1981: 91ff.) observes that annotated surface structures may be simply defined with respect to certain admissibility conditions (more on these shortly) without regard to an actual movement rule that maps from one level to another.[9]

Our goal, then, will be to assume that only annotated surface structure is built to test grammaticality.[10] We must now define more carefully just what annotated surface structure is in the current GB theory. We then show that these representations are at most linearly larger than their corresponding surface sentences.

We begin simply by describing the set of admissible annotated surface structures without reference to D-structure. That is, we define the set of annotated surface structures statically, in the manner that Joshi and Levy (1977) define a set of admissible tree structures. Roughly, the annotated surface structures of a given grammar are just the set of all well-formed labeled bracketings produced by the constraints of X theory plus the restrictions imposed by lexical subcategorization, plus bracketings where empty categories appear in certain positions, governed by a fixed set of conditions. In more detail, the well-formed annotated surface structures are defined inductively as follows:[11]

(I): Following standard assumptions, constraints along with locality conditions on subcategorization together yield a system describable by a context-free grammar (see, e.g., Gazdar and Pullum 1981). All NPs dominate some lexical material and correspond in one and only one way to the A positions, arguments subcategorized by the relevant verbs, again following the method outlined by Joshi and Levy (1977); the positions in English are: adjacent to the verb, for an object NP; first NP under S, for subject NP; first NP under PP for oblique PP, and so forth.[12] Further, all such lexical NPs must appear in argument (A) positions, where the notion of an argument position again depends in a strictly local way on the verb (e.g., the subject position of *seem* in English is not an argument position). Finally, we allow a finite number of specified lexical deletions (of particular words), such

[8] The details of the testing procedure are not given here, but may of course add some fixed space to the *kn* bound required to write down the annotated surface structures.

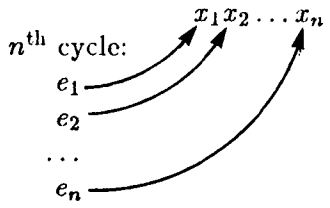
[9] At least, this seems to be so for all cases in English. But a note of qualification is required. There may be subtle examples showing that D-structure must be explicitly rebuilt in order to test grammaticality. Such examples do not seem to arise in English, but they may in other languages, such as Italian. So for example, it may be in Italian that the grammaticality of such examples as *was built a house* may demand explicit reference to D-structure, in order to determine whether a verb is a real passive or merely adjectival. If so, then the conclusions in the main text might not hold, since D-structure would have to be built.

[10] Note that this is true of the Marcus parser (Marcus 1982).

[11] Even this account is incomplete in some details, ignoring certain alternative formulations of the theory. But these defects can be repaired at the cost of adding more or slightly different clauses to the definition. For example, we omit a discussion of clitics, verb movement, government defined as mutual c-command, or Subject–Verb agreement. This last constraint may be defined via lexical insertion contexts, following Chomsky (1965) as formalized by Joshi and Levy (1977).

[12] Note that all these constraints are readily checked in the manner of Joshi and Levy (1977).

Case 1: successive elements of individual nodes
$n + 1^{st}$ cycle:



Case 2: adjunction
$n + 1^{st}$ cycle:

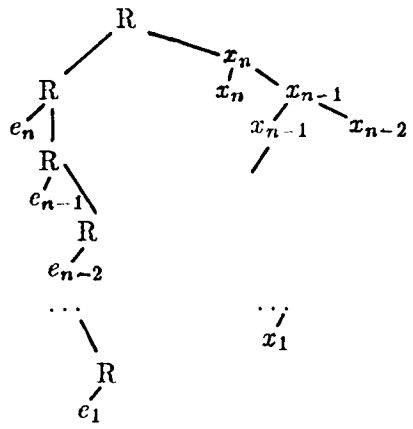


Figure 3. Successive cyclic trace-based Kimball analyses.

as *of, you*, in any single phrase, as long as no other constraints are violated. All labeled bracketings meeting these conditions are well-formed annotated surface structures.

(II): Any of the structures of (I) with empty categories (e.c.'s) replacing NPs, subject to the following conditions, are well-formed annotated surface structures:
- (i) Every such e.c. is an atomic constituent with a numerical index and with no internal bracketing;
- (ii) If the e.c is *governed* by some X^0 (lexical element such as verb, noun, and so on), where X governs Y iff the first branching node dominating X dominates Y, and there is no intervening maximal projection (full phrase) between X and Y then:
  1. the e.c. must be c-commanded by an NP antecedent(= element with the same numerical index), where c-command is defined just as government but dropping the clause about maximal projections; and
  2. the antecedent is either a lexical NP or another e.c. in a non-argument (A) position (the complement of the A positions defined above); and
  3. the e.c. must be subjacent to that antecedent, where subjacency is defined as usual.
- (iii) Else, the e.c. is ungoverned (is a "sc PRO") and can receive an arbitrary index.[13]

(III): Any of the structures defined by (I) and (II), and, in addition, with a *wh* phrase in COMP position c-commanding a governed e.c., or another *wh* phrase in COMP position and with the same index as that other e.c. or phrase, is a well-formed annotated surface structure.

(IV): Any of the structures defined by (I)–(III), and, in addition, with one of those structures with an e.c. having an index the same as that of an element adjoined to VP (following Baltin 1982), and c-commanded and subjacent to that element, is a well-formed annotated surface structure.[14] There can be at most one such adjoined position.

(V): Any of the structures defined by (I)–(IV) conjoined so as to meet Williams's (1978) **Across the Board** (ATB) conventions is a well-formed annotated

[13] Subject to constraints dictated by "control" theory, that is. We ignore this matter here by assuming an arbitrary index for PRO; this does not bear in any essential way on the description of the possible annotated surface structures. Neither c-command nor subjacency seem required for control; hence this may fall under whatever mechanism it is that interprets the indices of ordinary pronouns generally, a matter we leave outside scope of annotated surface structure.

[14] Note that the position so adjoined to VP is not part of the obligatory argument structure mentioned by the verb.

surface structure.[15] Without going into detail on the ATB constraint, its effect is to place e.c.'s in a conjunct if its verb or verb phrase is missing; the e.c. is bound to a c-commanding antecedent, as before. In the case of more complex reduced conjunctions (*The meat is ready to heat*, serve, and eat) the missing constituent sequence may be represented by a single e.c. in each missing position, though alternative analyses are to be preferred here.[15]

(VI): Nothing else is a well-formed annotated surface structure.

We also need some technical constraints. As pointed out above, we must assume that the actual index of an NP (as denoted by a subscript) takes up no extra space beyond the constant storage required for a distinct nonterminal name. Otherwise the amount of space required to write down an annotated structure of length proportional to n could be at worst proportional to $n \log n$, where the $\log n$ factor is used to hold the number of the index n. To sidestep this problem we assume a denumerable infinity of distinct NP "names".

This same assumption must be made explicitly or implicitly in any theory that assumes co-indexing but still strives for linearity in the size of underlying structures. Consider Kaplan and Bresnan's sketch (1982:263-267) that lexical-functional language recognition uses only linear space. In the full description of lexical-functional languages, names are distinguished as co-referential or not. Thus, two occurrences of, say, *Mary* must be distinct. In the LFG formalism, this is indicated by subscripts. (See for example Kaplan and Bresnan's examples 1982:225-227.) But then, this means that the sheer size of a functional structure (f-structure), the lexical-functional analog of an annotated surface structure, could be of size $n \log n$, again with $\log n$ space for the indices. Just writing down one f-structure could take more than linear space. Kaplan and Bresnan do not say in any detail just how they intend to check for sentence-hood using just linear space, but since all of their descriptions involve building an f-structure, we may assume that at least this much space will be required. In short, in order to a linear space bound, Kaplan and Bresnan need to adopt exactly the proposal made above.

A second key assumption is that traces may not be nested; a trace cannot contain another trace. This ban is required because otherwise we could build a tree representation containing just empty elements (the traces). Since a tree can be arbitrarily large, a single NP or S domain could have an arbitrarily large but surface-empty structure of elements, just what was to be avoided. We are now ready to state just what we want to show.

Theorem Let G be a government-binding grammar, and $L(G)$ the language it generates. Let AS_i be the annotated surface structure) associated with sentence w_i in $L(G)$. (If there is more than one such annotated surface structure, then AS_i is a set of annotated surface structures; AS_i is a singleton set if there is just one annotated surface structure.) Then there is a constant k such that for all sentences w_i in $L(G)$, and for all annotated surface structures AS_i underlying w_i, $|AS_i| \leq k|w_i|$.

The proof proceeds by induction on the number of cycles (S or NP domains) in an annotated surface structure corresponding to a sentence in $L(G)$. First we shall show that the length of a one-cycle annotated surface structure is linearly proportional to its corresponding surface sentence. This will be easy, since within a single cycle (S or NP domain), there can be movement to at most a fixed number of "landing sites" as defined above: the $\overline{A}$ positions, plus COMP, plus one adjunct to a VP. The lexical entry for a verb mentions only a finite number of such arguments. The one additional landing site adjoined to VP can receive only one phrase, because in order to receive more, additional phrases would have to be adjoined in the manner of the Kimball-type structures discussed in the previous section. But these would violate subjacency.[16]

Once we have established linearity in the base case, we now look at annotated surface structures i and $i+1$ cycles deep. Assuming that structures of depth i maintain linearity, we show that those of depth $i+1$ do also. This step is tedious, since one must go through the possible ways to obtain the $i+1$ cycle from the one preceding it, one by one. The landing site analysis is exploited here, as is subjacency. The empty category analysis is also used. Subjacency helps because there is no way to "skip" a cycle, constructing structures of depth $i+2$ from those of depth i directly.

Proof

Basis step. $i = 1$ (bottom cycle, no embedded sentences or NPs.) Given a surface sentence w_i, we consider the length of the corresponding annotated surface structure. Let s = the length of the surface sentence. There are four cases.

Case 1. No e.c.'s in the S or NP cycle, and no specified lexical deletions. Assume a context-free base with no useless nonterminals or cycles, and with rules where the length of the longest righthand side is p. If m = the number of nonterminals in the derivation of a sentence in this grammar, then $m \leq cs$ for some fixed positive integer c, as may be easily verified by induction. In addition, to write down the annotated surface structure, we must add two bracket labels for each nonterminal symbol. Thus $|AS_i| = 2m + s \leq 3cs$. Note that if we wanted to establish a relationship between debracketed annotated surface structures and surface strings, then this last step would be unnecessary.

Case 2. A finite number of specified lexical deletions within this cycle, e.g, *of*, as in, *all of the people* → *all the people*, or an imperative (if a root sentence). Let the

[15] For a more recent formulation of the ATB conventions as the linear union of phrase markers, see Goodall (1983). We note in passing that the phrase marker union also preserves linearity of annotated surface structures.

[16] Recall that now we are applying subjacency as a static constraint on annotated surface structures. In fact, since in the basis step we consider only annotated surface structures one S or NP cycle deep, this case does not arise.

maximum number of these deletions be K. Then $|AS_i| \leq 3cs + 3K$. For $s \geq 1$, $3cs + 3K \leq 3cs + 3Ks = (3c+3K)s$. Let $c' = 3c + 3K$. Then $|AS_i| \leq c's$. Again, we can omit the $2K$ factor for the debracketed case.

Case 3. Empty categories within this (S or NP) cycle, with antecedents in the same S or NP. There are a finite number of such positions, as described earlier: only NP argument positions (thematically marked by the verb); or the adjoined position to VP. Let C bound this number from above. Then $|AS_i| \leq c's + C$; using the same approach as in case 2, the righthand side of this inequality is less than $c''s$. Clearly, combinations of cases 2 and 3 cause no problems because we can add a constant number of deletions together with the constants obtained from within cycle e.c.'s to obtain a new constant factor.

Case 4. Empty categories in the cycle without antecedents in that domain. If an empty category exists in an S or NP without an antecedent (NP, *wh*, etc.) in that domain, then clearly the corresponding surface string is shorter than the corresponding annotated surface structure, since it does not include the empty category symbols. However, again the addition of each empty category symbol adds just one to the total annotated surface structure length, and there are at most a finite number of such positions ($\overline{A}$ positions, such as COMP, as described earlier). Therefore, the corresponding annotated surface structure is just a constant longer than the corresponding surface sentences, as in Case 3.

Well-formed annotated surface structures exhibiting the features of gapping, VP deletion, and conjunction reduction do not show up at this step, since they combine two i level cycles into an $i+1$ domain. They are considered in the induction step. This completes the basis step.

Induction step. Suppose that up through cycle i we have that $|AS_i| \leq ks_i$, where s_i is the terminal length at the ith cycle, and k is a constant. We now must show that this relation holds for structures of depth $i+1$. There are five possibilities.

Case 1. No empty categories athe top level of cycle $i+1$. Then the terminal string associated with this cycle consists of two parts, whatever terminals are introduced directly by nonterminals in cycle $i+1$ and new elements of cycle $i+1$ bound to e.c.'s in cycle i. But there are a finite number of empty category sites for material in the current domain, by the definition of a well-formed annotated surface structure. Call this number C. By the inductive hypothesis, any of these constituents themselves meet the condition that their annotated surface structures are bounded above by a linear multiple of their terminal strings. Thus the total annotated surface structure for the current cycle is at most C times the bound on previous cycles, plus a constant to accommodate the length of terminals introduced directly in cycle $i+1$.

$$AS_{i+1} \leq d\Sigma_{j=1}^{i} AS_j + d'$$

$$\Sigma_{j=1}^{i} AS_j \leq C \Sigma_{j=1}^{i} s_j$$

Substituting, we obtain:

$$AS_{i+1} \leq C \Sigma_{j=1}^{i} s_j + k''s_{i+1}$$

$$\leq ks_{i+1}$$

Case 2. Specified deletions in cycle $i+1$. If there are a finite number of specified lexical deletions, this is just like the basis case. This case includes the introduction of PROs. PRO can appear in a finite number of new positions in cycle $i+1$ (the Subject position, if ungoverned).

Case 3. E.c.'s with antecedents within cycle $i+1$. The demonstration proceeds as in the basis case.

Case 4. E.c.'s with antecedents in cycle $i+2$. Again, like the basis case. This cannot change the linearity bound.

Case 5. Annotated surface structures with empty verb, verb phrase, and coordinate reduction positions. This is the only new situation that arises in the induction step as opposed to the basis step. Suppose we have a conjunct formed by deleting material from each of n conjuncts. An example is *the meat is ready to take out of the fridge, heat, and serve.* The example is from Rounds (1975:137) attributed to E. Bach. If such constructions involved actual recovery of deleted deep structure material, then problems could arise. The literal material would have to be copied, and we could get a linearity-violating Peters-type sentence.

But this problem can be avoided with an interpretive approach governed by the "across the board" conventions of Williams (1978). We supply indices, not actual copied material, for the well-formed annotated surface structure. The ATB constraint lines up the conjuncts to be co-ordinated, one under the other. For example, a sentence like *the meat is ready to heat, serve, and eat* is factored as follows, where we have deleted duplicate material in other conjuncts.

The meat$_i$ is ready	to heat	e_i
1	serve	3
	eat	
	2	

We can represent term (2) as an unordered set of lexical items, for example, *heat, eat, serve.* Plainly, this representation cannot be more than linearly larger than the surface sentence.[17]

Similar results hold for empty categories linked to verbs and verb phrases. Each cyclic domain of the the associated annotated surface structure contains a constant number of empty VP "gaps", denoted [e]; there can be at most one main verb, VP, or auxiliary verb sequence

[17] Again, the Goodall (1983) representation would be suitable here.

per cyclic domain. Therefore, the total number of gaps in the conjoined structure is bounded from above by a constant times s_{i+1}, the length of the terminal string.

This exhausts the range of possible cases, completing the induction and the proof.

The linearity demonstration shows restricting deletions has a powerful effect on the weak generative capacity of a transformational grammar. The implications of that result for linguistic description are discussed in the next section.

3. The Root of Complexity: Lexical-Functional Grammars and GB grammars

The results of the previous section show something about the weak generative capacity of modern transformational grammars. The study of weak generative capacity is not an end in itself, however. In the best case, we would like weak generative capacity to be a kind of diagnostic aid to tell us that something is amiss with a linguistic theory. We would like our theory to be able to describe all and only the natural languages. A theory could fail to do this in two ways, either in terms of weak generative capacity or in terms of strong generative capacity. A theory that is too powerful could generate either unnatural tree structures (and so be too powerful in terms of strong generative capacity) or it could generate unnatural sentences (and be too powerful in terms of weak generative capacity). If we are interested in the rule systems (grammars) that underly linguistic behavior, then it is ultimately strong generative capacity that is of interest. Still, weak generative capacity can help here to point the way to excess strong generative capacity. We will also not want to stop at diagnosis. We also want to determine just *why* a particular theory can generate too many languages – what the source of its excess power is. We saw that with *Aspects* transformational grammars the additional power lay with unbridled deletion. What of other recent theories of grammar?

In this section we shall present an example of exactly this kind. This will be a language that is presumably not a natural language. We will use this language as a "probe" into the power of current linguistic theories. We shall see that this language can be easily generated by lexical-functional grammar, but not by a GB grammar. More important, this weak generative result has a strong generative capacity reflex. We can use this result to locate the excess power of the LFG system. This could be of value in discovering restrictions for the LFG system. In terms of strong generative capacity, the more important goal, we shall see that the LFG theory has the ability to define unification predicates over hierarchical tree structures, something unavailable in the GB theory. This extension of the traditional definition of linguistic predicates has implications for the ability of LFG to describe unnatural grammars, not just unnatural languages.

Here is what we mean to show in more detail. LFGs use a particular kind of unification machinery (described below) in order to account for well-formed sentence structures of Dutch (Bresnan, Kaplan, Peters, Zaenen 1982). This unification procedure is central to the construction of the grammatical structures of lexical-functional theory. But it is also powerful enough to describe grammars quite unlike any natural grammatical system. By changing the Dutch LFG only slightly we can produce a rule system that allows "object control" via a preceding NP (as in *Mary persuaded John to leave*) just in case the NP in question and the controlled position are equally deeply embedded. This we take to be an unnatural rule system.

To begin, we present our artificial "diagnostic" language, the power of 2 language, $L_2 = \{a^i | i \text{ is a power of } 2\}$. L_2 is a lexical functional language, since the following lexical-functional grammar generates it:

1. $A \rightarrow \underset{(\uparrow f) = \downarrow}{A} \quad \underset{(\uparrow f) = \downarrow}{A}$

2. $A \rightarrow \underset{(\uparrow f) = 1}{a}$

The $(\uparrow f) = \downarrow$ functional structure constraints on the nonterminals enforce the restriction that the same number of A expansions be taken on each subtree; expansions are symmetric all the way down the "words", the *a*s. This guarantees a power of 2 expansion; the details are left to the reader.

We can now ask deeper questions. First, *why* can lexical-functional grammars generate such languages? More on this shortly. Second, can L_2 be generated by a GB grammar? To answer the second question first, the answer here seems to be no, because of a property of GB languages that is violated by L_2, namely, the **constant growth property**, defined and discussed for tree adjunct grammars by Joshi (1983). This property will only be briefly explored below; for more complete remarks, see Berwick and Weinberg (1984).

If we arrange the sentences of L_2 in order of increasing length, we see that they become farther and farther apart. In fact, for any fixed set of constants C, we can always find a sentence of L_2, w_i, say, such that there is no w_j in L_2, with $|w_i| = |w_j| + c$, for $c \in C$. We state this property as follows:

Definition. A language L is said to possess the constant growth property (or be constant growth) if and only if there exists a constant M and a set of constants C such that for all sentences $w_k \in L$ with $|w_k| > M$, there exists another sentence in L, w_k', such that w_k is at most a constant longer than w_k', $|w_k| = |w_k'| + c$, for $c \in C$. A grammar is said to possess the constant growth property iff the language it generates is constant growth.[18]

Lexical-functional grammars, then, are not constant growth. In contrast, government-binding grammars cannot generate such languages because they are constant growth. Intuitively, the demonstration works much like the linearity proof. For a full discussion, see Berwick and

[18] So far as it can be now determined, constant growth seems to be a purely mathematical property of natural languages that has no clear "functional" reflection. Presumably, constant growth is a derivative of other, deeper properties of natural languages.

Weinberg (1984: Appendix A). The point is that no government-binding grammar can generate L_2 or any nonconstant growth language.[19]

What is it that gives lexical-functional grammars their ability to define languages like L_2? LFGs can test complete subtrees for compatibility. At a dominating node we can check whether an entire hierarchical structure is feature compatible with another structure. This follows from the account of functional structure unification defined by Kaplan and Bresnan (1982). Functional structures are hierarchical in nature; they are directed, acyclic graphs. Functional structure well-formedness is defined by the condition of functional structure uniqueness. Roughly speaking, there can be no conflicts in the assignment of feature complexes, even if those features are in fact hierarchical structures.

This kind of feature compatibility test goes well beyond that required for the checking of "ordinary" agreement, as in subject-verb number agreement. When we test a subject and verb for agreement, all that we do is check an unordered list of features for compatibility. The number, gender, and so forth of the subject NP must agree with that of the verb, as percolated through the VP. It is a far cry from this kind of agreement checking to the "agreement" of two entire tree structures, but this is what is implied by the lexical-functional unification procedure.[20]

As we saw in our earlier example, this unification procedure is sufficient to generate the power of 2 language. A natural question to ask then is whether this ability to compare entire functional structures is *necessary*. For if all cases of functional structure unification can be replaced by unordered feature agreement tests, then there is no motivation for adopting the more powerful mechanism, at least on these grounds.

The lexical-functional theory is, perhaps, already committed to the ability to test hierarchical functional structures for compatibility. For functional structures are certainly hierarchical in nature. They must encode the hierarchical relationships between root and embedded propositions, for example. A functional structure is used as the input to semantic interpretation, and so must reflect hierarchical dependencies. Otherwise we cannot decipher the relationships in a complex sentence like *John expected Mary to persuade Bill to win.* The feature checking machinery must be designed to test for functional structure compatibility because that is the only level of representation where features like the number of the subject are to be found. But once we permit feature checking of functional structures at a single, unembedded level for the number of a subject NP, it is hard to see how we can rule it out for a more complex functional structure.

In fact, lexical-functional researchers have proposed natural language cases where one must check one complex functional structure for compatibility against another. Just such a case has been discussed by Bresnan, Kaplan, Peters, and Zaenen (1982), in the analysis of certain Dutch sentences. We will not review all the details of their proposal here except to establish the point that hierarchical functional structure comparisons are crucially implicated. The data Bresnan et al. want to account for is this. Dutch contains infinitely many sentences of the following sort (examples from Bresnan et al. 1982: 614):

. . . dat Jan de kinderen zag zwemmen
. . . that Jan the children saw swim
. . . that Jan saw the children swim

. . . dat Jan Piet Marie de kinderen zag helpen laten zwemmen
. . . that Jan Peter Marie the children saw help make swim
. . . that Jan saw Piet help Marie make the children swim

These Dutch sentences must have a certain constituent structure. Their proposed structure consists of two branching "spines", one a right branching tree of VPs containing objects and complements, the other a right branching tree of V containing verbs without their objects and complements. Every verb uses its lexical argument structure to demand certain NP objects or that the verb complement's subject be controlled by the verb's object or subject. For example, the verb *zag* demands that its object control the subject of *zag*'s verbal complement. This is analogous to the English case where the object of a verb, for example, *persuade*, controls the subject of *persuade*'s complement, as in, *We persuaded John to leave.*

The lexical-functional system encodes this agreement in number of verbs and NPs by forcing an identification between the functional structure of the object of *zag* and the functional structure of the verb complement of *zag* (denoted VCOMP). The "equation" is written (↑ VCOMP SUBJ) = (↑ OBJ). The problem, of course, is that if we have three verbs then we have three such constraints, but the associated NPs that satisfy them lie along a distinct VP "spine" of the constituent structure tree that is separated from the verbs along the $\overline{V}$ spine. In other words, the "control" equations are built up along the rightmost, $\overline{V}$ spine of the constituent structure tree, but the NPs that satisfy these equations lie along the left side. How can we assemble the NP functional structures for proper checking against the control equation demands? Because feature checking can occur only at some common dominating mother node, the first place where all elements are "visible" to each other is at the first VP node completely dominating both right and left subtrees. The way that Bresnan et al. accomplish this task is to build up along the rightmost subtree a functional structure representation that encodes all of the control equations, in the form of a hierarchical functional structure with unfilled slots for the subjects and objects mentioned by the controlling verbs. Note that the structure is indeed hierarchical, containing embedded components. Along the lefthand side of the constituent tree Bresnan et al. build up a second hierarchi-

[19] Another example is the language of perfect squares.

[20] Several other theories also adopt a directed, acyclic graph notation for features, among these, Kay's (1982) unification grammar and Shieber's (1983) PATR II formalism. Interestingly, Sag et al. (1984) adopt the more restricted view of features.

cal functional structure that "merges" successfully into the righthand one just in case the number of NPs and their assignment to controlled positions meshes with the "slots" left remaining in the righthand functional structure.

One must build and check a **hierarchy** of features because in order to encode the possibility of an arbitrary number of controlled NPs below the dominating VP node, we must adopt some means of encoding a potentially arbitrary number of features (denoting each of the NPs and their associated verbs). But given that the functional structure "equations" annotating the underlying context-free grammar are fixed once the grammar is written down, the only way to do that is by building up some recursive structure that mimics the constituent structure derivation as a chain. (With only a finite number of features, we can only encode an infinite number of different cases by means of chains or trees.) This means that Bresnan et al. are forced to adopt hierarchical feature checking as the means to describe the Dutch sentences.

In contrast, the government-binding theory represents the same pairing of NPs and verbs via a "flat" co-indexing scheme. *Jan de kinderen zag zwemmen* would be roughly, $NP_1\ NP_2\ [_{\overline{V}}\ V_1\ V_2]$ in annotated surface structure (see Evers 1975 and Berwick and Weinberg 1984). As outlined in Berwick and Weinberg (1984), potential co-indexings can be evaluated by non-erasing pushdown transductions that test only single, unanalyzed nodes, never building up tree-structured features as in the lexical-functional grammar example. (Note again that D-structures are not reconstructed to carry out this check.)

The problem for the lexical-functional machinery is that once hierarchical checking is admitted for this one example, there is nothing to bar it in other cases. But then the power of 2 language can be generated. One can also build "unnatural" lexical-functional grammars using just the linguistically motivated control equation apparatus and phrase structure rules proposed in the lexical-functional theory. The same linguistically motivated rules used for Dutch, combined in slightly different ways, lead to grammars quite unlike anything ever attested or likely to be attested in natural rule systems. The example we give uses almost precisely the Dutch control equations, and a slightly different context-free base.

The idea behind our unnatural grammar is this. We will build a grammar where a verb controls a higher object NP just in case both the verb and that NP are essentially equally deeply embedded along different "spines" of the constituent structure tree. This we take to be a highly unnatural system. There is no natural language where a control property "counts".

We need these context-free rules and their functional structure annotations:

1. VP → NP (↑ subj) = ↓ V (↑ Vcomp) = ↓ ($\overline{V}$) ↑ = ↓
2. VP → NP (↑ obj) = ↓
3. $\overline{V}$ → V (↑ Vcomp) = ↓ V
4. $\overline{V}$ → V
5. NP → N

Rules (2)–(5) are precisely those used by Bresnan et al. Rule (1) is different. (1) has the associated equation (↑ subj) = ↓ attached to the NP node instead of the equation (↑ obj) = ↓. We must also add new lexical entries for the following "verbs":

V3: (↑ Vcomp subj) = (↑ obj)
(↑ pred) = $V_3\langle$(↑ subj)(↑ ob)(↑ Vcomp)$\rangle$
V2: (↑ pred) = $V_2\langle$(↑ subj)(↑ Vcomp)$\rangle$
V1: (↑ pred) = $V_1\langle$(↑ subj)(↑ Vcomp)$\rangle$

The effect of this modest change is a rule system that has exactly the properties we claimed. Consider first the functional structure built up along the lefthand VP branching spine. The last NP expansion will have the associated equation (↑ obj) = ↓. Each VP demands that the VCOMP functional structure component associated with the node above it be identified with the functional structure built up at that VP. The effect is to build up a hierarchical arrangement of VCOMP functional structures, one for every VP node that is generated except for the top and the bottommost VP. In addition, a subject functional structure component is passed up from all NPs but the last one.

The object from the lefthand functional structure merges into this righthand structure successfully if and only if it has one level of embedding less than the righthand structure. This is our desired result. Otherwise the object structure cannot be laid on top of the righthand structure and overlap properly; it must coincide with the empty object slot on the righthand side.[21]

4. The Formal Characterization of Natural Languages

Summarizing the analysis so far, we have seen just how modern transformational theories differ formally from their older counterparts. We have also seen that that difference is reflected as a weak and strong generative capacity difference between the new theory and the lexical-functional theory.

Some questions are still unanswered. In the previous section, we came to a partial diagnosis of the source of the extra power of the lexical-functional theory. In this section we would like to pin down that diagnosis. At the same time we shall offer a different perspective on the formal characterization of natural languages. This analy-

[21] For example, suppose we interchanged V_2 and V_3. Then the control verb V_3 is less deeply embedded than the object NP it is supposed to control. This structure should be ruled out, and it is. The lefthand functional structure will be as before. But now the righthand functional structure will not merge properly with the lefthand functional structure because that functional structure demands that the object be embedded inside two VCOMPs, whereas the righthand structure calls for an object embedded inside just one. Similarly, if V_3 were embedded one more level down, the number of VCOMPs would not match. Only when the number of embeddings is the same (plus one) on both left- and righthand sides is the structure well-formed.

sis will necessarily be more speculative. Still, it is hoped that the discussion will provoke a fresh look at how to go about the mathematical analysis of natural languages.

To begin, let us recall that the suspected source of extra power in the lexical-functional theory is the unification procedure defined over hierarchical structures (constituent structures). We also argued that nothing like this kind of power is required to describe natural languages. In this section we shall investigate this claim more deeply. We shall look at one case, co-ordination, that might seem to require full unification, and see that in fact hierarchical unification is not required.

At first glance, co-ordination would seem to demand some kind of unification predicate. The reason is that co-ordinate structures obey a familiar principle (Williams 1978) that permits only "similar" conjuncts to be linked.[22] One way to visualize the parallelism constraint is to imagine the two conjuncts being laid on top of one another. If they match, then the conjunction is permitted, otherwise, it is not permitted. Williams (1978) formalizes this condition. This process is reminiscent of the lexical-functional unification procedure. (Compare it to the Dutch example given earlier.) Here too, we "overlaid" two hierarchical structures to determine well-formedness. The Dutch sentences were legal just in case two hierarchical spines could be so overlaid, or unified. Is unification required?

On closer inspection the analogy with unification breaks down. It is true that the parallelism of co-ordinate conjuncts demands a match in terms of phrasal nodes. The key difference between lexical-functional unification and the co-ordination constraint is that co-ordinate parallelism need only hold at the *top level* of a phrasal sequence. Internal details of the matched conjuncts do not matter. This is in contrast to the unification predicate, which, as the Dutch example shows, can demand a hierarchical match. For example, the following conjunction is perfectly grammatical, even though the conjoined VPs are internally different, one containing an Adjectival Phrase and the other a Noun Phrase (example from Goodall 1983): *the bouncer was muscular and was a guitarist.* One can even conjoin active and passive sentences (*John went to Boston and was taken for a ride*). As Goodall (1983) demonstrates, one way to describe this effect is as the union of the top level of phrasal nodes (actually, phrase markers).

In contrast, the Kaplan and Bresnan unification procedure (1982: 272), as defined by their statement (190c), recursively defines a union over what may be an entire tree:

> (190) c. If e_1 e_2 are both f-structures, let A_1, A_2 be sets of attributes e_1 and e_2, respectively. Then a new f-structure e is constructed with $e = \{\langle a,v\rangle \mid a \in A_1 \cup A_2$ and v = merge [Locate $|(e_1,a)]$, Locate $[(e_2,a)]]\}$ (*Locate* is an operator that actually finds the sub-f-structure with the specified attribute structure.)

Here, $\langle a, v\rangle$ is the union of a *hierarchical* attribute set, since this last step is carried out recursively to all levels of structure. This means that there is nothing to stop us from writing a co-ordination rule in the lexical-functional system that demands equality in tree structure through all levels of hierarchical detail, contrary to what is observed.[23] We might speculate then that a *general* property of constraint statements in natural languages is that they are defined in terms of predicates on linear sequences of structures (phrase markers), rather than by hierarchically defined unification predicates. It remains to explore just what this restriction comes to, but it is clear that this is exactly where and how lexical-functional grammar diverges from the "classical" view of generative grammar. The classic view, outlined in Chomsky's *Logical Structure of Linguistic Theory*, defined predicates in terms of a concatenative algebra at each of several levels of representation (phonetic, syntactic, and so forth). The details are not essential here, but one property of these algebras is: they fixed predicates in terms of linear sequences of elements, rather than trees.[24] The lexical-functional system extends the power of representational description to include the possibility of unification predicates defined over nonlinear constituent structures. While this violation of the usual syntactic adjacency restrictions (observed from earliest days of generative grammar) is certainly sufficient to describe natural languages, the examples presented here show that it is not necessary.

This diagnosis also tells us one way to repair the lexical-functional theory. One could restrict the lexical-functional theory to ban hierarchical unification predicates. One way to do this is to simply eliminate the recursive step of Kaplan and Bresnan's unification procedure (190c), (1982:272) excerpted earlier. For example, feature merger could be restricted to operate over just two cyclic (S or NP) domains. One would still need a way to handle constructions like those in Dutch, or, should they be necessary, the *ww* constructions. Of course, it may be that other restrictions suffice.

Whatever the outcome of these changes, a more general question for future work centers on the status of the concatenation algebras underpinning traditional generative grammar. While there has been some formal work in this area (see Borgida 1983 and Berwick 1982), it remains to be seen whether the linear predicates presupposed by such a model do indeed characterize what it means to be a natural grammar. If they do, then extensions to more general unification predicates, as in LFG, unification grammar, or PATR-II, may well be unwarranted.

Acknowledgments

Much of this research has been sparked by collaboration with Amy S. Weinberg. Thanks to her for many discussions on GB theory. Portions of this work have appeared in *The Grammatical Basis of Linguistic Perform-*

[22] See Sag et al. (1984) for a different formulation of "similar".

[23] This is true also of *respectively* type constructions: evidently, only the linear union of phrase markers is required to define these sentences, not the tree-checking power used by Kaplan and Bresnan (1982: 269-271).

[24] This is true even of linear phrase markers systems that directly admit discontinuous constituents, which Chomsky's system does not; these include McCawley's (1982) proposal and Higgenbotham's (1983) recent elaboration of McCawley (1982).

ance. The research has been carried out at the MIT Artificial Intelligence Laboratory. Support for the Laboratory's work comes in part from the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014–80–C–0505.

References

Baltin, M. 1981 A Landing Site Theory of Movement Rules. *Linguistic Inquiry* 13: 1-38.

Berwick, R. 1982 Locality Principles and the Acquisition of Syntactic Knowledge. PhD dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, Massachusetts.

Berwick, R. and Weinberg, A. 1982 Parsing Efficiency, Computational Complexity, and the Evaluation of Grammatical Theories. *Linguistic Inquiry* 13: 165-191.

Berwick, R. and Weinberg, A. 1984 *The Grammatical Basis of Linguistic Performance.* MIT Press, Cambridge, Massachusetts.

Borgida, A. 1983 Some Formal Results about Stratificational Grammars and Their Relevance to Linguistics. *Mathematical Systems Theory* 16: 29-56.

Bresnan, J. and Kaplan, R. 1982 Introduction: Grammars as Mental Representations of Language. In: Bresnan, J., Ed., *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge, Massachusetts: xvii-lii.

Bresnan, J.; Kaplan, R.; Peters, S.; and Zaenen, A. 1982 Cross-serial Dependencies in Dutch. *Linguistic Inquiry* 13: 613-636.

Chomsky, N. 1955 *The Logical Structure of Linguistic Theory.* Plenum Press, New York, New York, 1975.

Chomsky, N. 1981 *Lectures on Government and Binding.* Foris Publications, Dordrecht, Holland.

Evers, A. 1975 The Transformational Cycle in Dutch and German. PhD dissertation, Department of Linguistics, Rijksuniversiteit, Utrecht, Holland.

Ginsburg, S.; Greibach, S.; and Harrison, M. 1967 One-way Stack Automata. *Journal of the Association for Computing Machinery* 14: 389-418.

Goodall, G. 1983 Coordination. Unpublished draft of thesis, University of California at San Diego.

Higgenbotham, J. 1983 A Note on Phrase Markers. *Revue Québécoise de Linguistique* 13(1): 147-166.

Hopcroft, J. and Ullman, J. 1979 *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachusetts.

Hornstein, N. 1984 *Logic as Grammar.* MIT Press, Cambridge, Massachusetts.

Johnson-Laird, P. 1983 *Mental Models.* Harvard University Press, Cambridge, Massachusetts.

Joshi, A. 1983 Some Formal Results about Tree Adjunct Grammars. *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics.*

Joshi, A. and Levy, L. 1977 Constraints on Local Transformations. *SIAM Journal of Computing* 6: 272-284.

Kaplan, R. and Bresnan, J. 1982 Lexical Functional Grammar: A Formal System for Grammatical Representation. In: Bresnan, J., Ed., *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge, Massachusetts: 173-281.

Kay, M. 1982 Unification Grammar. Xerox PARC unpublished ms.

Kimball, J. 1967 Predicates Definable by Transformational Derivations by Intersection with Regular Languages. *Information and Control* 11: 177-195.

Lapointe, S. 1977 Recursiveness and Deletion. *Linguistic Analysis* 3: 227-265.

Marcus, M. 1982 *A Theory of Syntactic Recognition for Natural Language.* MIT Press, Cambridge, Massachusetts.

McCawley, J. 1982 Parentheticals and Discontinuous Constituent Structure. *Linguistic Inquiry* 13: 1.

Peters, S., 1973 On Restricting Deletion Transformations. In: Gross, M.; Halle, M.; and Schutzenberger, M., Eds., *The Formal Analysis of Language.* Mouton, The Hague, Holland: 372-384.

Peters, S. and Ritchie, R. 1973 On the Generative Power of Transformational Grammars. *Information Sciences* 6: 49-83.

Pullum, G. 1984 Syntactic and Semantic Parsability. *Proceedings of Coling84,* Stanford, California: 112-122.

Rounds, W. 1975 A Grammatical Characterization of the Exponential Time Languages. *Proceedings of the 16th Annual Symposium on Switching and Automata Theory:* 135-143.

Salomma, A. 1971 The Generative Capacity of Transformational Grammars of Ginsburg and Partee. *Information and Control* 18: 227-232.

Sag, I.; Gazdar, G.; Wasow, T.; and Weisler, S. 1984 Coordination and How to Distinguish Categories. CSLI Report CLSI-84-3.

Shieber, S. 1983 Notes on the PATR-II formalism, SRI International, Menlo Park, California.

Thiersch, C. 1978 Topics in German Syntax. PhD dissertation, MIT Department of Linguistics and Philosophy, Cambridge, Massachusetts.

Williams, E. 1978 Across-the-board Rule Application. *Linguistic Inquiry* 9: 31-43.