Optimal Growth in Phrase Structure

David P. Medeiros

This article claims that some familiar properties of phrase structure reflect laws of form. It is shown that optimal sequencing of recursive Merge operations so as to dynamically minimize c-command and containment relations in unlabeled branching forms leads to structural correlates of projection. Thus, a tendency for syntactic structures to pattern according to the X-bar schema (or other shapes exhibiting endocentricity and maximality of 'non-head daughters') is plausibly an emergent epiphenomenon of efficient computation. The specifier-head-complement configuration of X-bar theory is shown to be intimately connected to the Fibonacci sequence, suggesting connections with similar mathematical properties in optimal arboration and optimal packing elsewhere in nature.

Keywords: c-command; minimalism; phyllotaxis; projection; X-bar theory

1. Introduction

This article addresses some theoretical issues in language design, adopting the biolinguistic concerns of the Minimalist Program (Chomsky 1995b). Within this framework, the line of inquiry pursued here is the attempt to explain linguistic properties in terms of 'laws of form' that may have nothing in particular to do with language, or even with biology, but rather seem to be at work at the deepest level in nature. Much has been written elsewhere clarifying and defending this sort of approach; see Chomsky (2005), Freidin & Vergnaud (2001), Uriagereka (1998), and Boeckx & Piattelli–Palmarini (2005), among others.

Within the Minimalist Program, much attention has been given to 'virtual conceptual necessity', and the intuition that 'that which is necessary is also sufficient'. As a result, one prominent trend in minimalist explanation is to reduce linguistic properties to requirements for 'legibility' with respect to the cognitive systems with which the linguistic system interacts (so-called 'bare output conditions'). Nevertheless, it deserves to be emphasized that various linguistic

I would like to thank the following individuals for help at various stages of this project: Andy Barss, Tom Bever, Cedric Boeckx, Andrew Carnie, Noam Chomsky, Sandiway Fong, Heidi Harley, Norbert Hornstein, Scott Jackson, Simin Karimi, Terry Langendoen, Terje Lohndal, Jaime Parchment, Massimo Piattelli–Palmarini, Sumayya Racy, Yosuke Sato, Juan Uriagereka, and participants in the University of Arizona Syntax Salon. I would also like to thank two anonymous reviewers for their feedback and advice.

biolinguistics
© The Author (2008)

ISSN 1450–3417

properties can be both real and subject to minimalist explanation without being required in order for language to work at all, or in the simplest possible way. One of the most important lessons from applying 'Galilean' thinking to the natural world is that sometimes the best (i.e. natural) solution is not the simplest. Often more than one constraint must be satisfied by a system in some optimal way, and when the constraints conflict, interestingly complicated structure may emerge. In language as well, the biolinguistic viewpoint leads us to expect to find certain properties that are more complicated than would be strictly required for language to work *at all*, but are nevertheless 'natural' if language works *optimally*.

1.1. Where We Are Headed

I propose in this article that certain properties of phrase structure have this kind of explanation, following not from bare output conditions but rather emerging 'for free' from concerns of efficient computation.[1] In particular, I propose here that the characteristic shape of phrases, as captured by the X-bar schema and similar forms, constitutes what we might think of as an 'optimal packing solution' or an 'optimal growth mode'. On the barest assumptions, Merge may apply freely to recursively build structure from terminal elements in any number of ways. However, if this implicitly free structure-building is subject to a constraint on efficient computation (related to minimizing computation involving c-command and containment relations), then some constructional choices will be preferred over others. Given basic concerns of locality of information flow in the derivation, it is plausible that this will induce certain consistent patterns in recursion (what amount to repeated structural 'templates'). Enumerating all possible recursive templates and comparing them with respect to this computational constraint, I show that the best templates have the shape of generalized X-bar projections. That is, the best way to 'pack' terminals into an iterated molecule of recursive structure (the best phrasal template) places a unique terminal at the bottom of the phrasal template, with 'slots' for several more objects of the same shape as the full 'phrase'. This kind of format is represented in (1).

(1) $[\ \alpha\ [\ \beta\ \ldots\ [\ \gamma\ [\ X^0\ \ \delta\ ]\ ]\ \ldots\ ]]$

As I will show, such a pattern of recursion produces fewer c-command and containment relations than any pattern of comparable complexity, a fact that I take to indicate computational optimality (e.g., minimizing the space searched by repeated probe-goal operations). In (1), X^0 is a terminal element, and α, β, γ, and so on are themselves constructed according to the pattern in (1). This is really shorthand for a class of optimal patterns, differing among themselves in how many self-similar 'slots' (α, β, γ, …) they permit. This includes (2), (3), and (4): In familiar terms, (2) corresponds to the geometry of the head-complement pattern, (3) to the specifier-head-complement pattern of the X-bar schema, and (4) to a

[1] As will be familiar to connoisseurs of this enterprise, the idea is that explanation for linguistic properties can fruitfully be pursued in terms of the abstract derivation that generates expressions. The relevant sense of efficiency is to be understood as internal to this abstract computation, rather than directly reflecting online processes in language use.

pattern in which every 'phrase' may have two 'specifiers'.

(2) [X^0 α]

(3) [α [X^0 β]]

(4) [α [β [X^0 γ]]]

We may describe the family of growth patterns fitting (1) as 'projective'. Some further factor(s) must act to select one particular choice (e.g., (3) instead of (4)) from the spectrum of projective solutions described by (1), a matter to which I return.[2] On the other hand, the format of (5) is not projective in the appropriate sense (because the terminal element X^0 is not at the 'bottom').

(5) [X^0 [α β]]

I believe this result is surprising and significant. The options for structure-building allowed here are quite free; any finitely-defined scheme incorporating terminals into indefinitely recursive patterns is considered. Needless to say, only a small minority of these patterns 'look like' projections. Other possibilities have a repeating phrasal template which places terminals at (potentially many) designated locations other than the 'bottom', or recurse via units different than the 'top' of the template, and so on.[3] The considerations which enter into the investigation are of a purely configurational, geometric nature; no notion of 'head of a phrase', 'label', or other elements of the theory of projection are built into the assumptions. Yet something akin to projection (more precisely, a structural basis which could readily be mapped to a projection scheme) emerges 'for free' as an optimal solution. This suggests that the property of projection may be an epiphenomenon of 'blind' structural optimization.

A final point worth mentioning here is that what is explained is an optimal tendency, not an absolute law. As is the case with laws of form more generally, this kind of explanation is actually strengthened by finding occasional deviations from the predicted pattern (so long as they are rare). Consider, for example, the pervasive Fibonacci pattern in plant growth. A certain species may display this pattern as an overwhelming tendency, but individuals may show other patterns (or, as often happens, a deviation from the pattern is found on one portion of a single individual otherwise adhering to the pattern). In such cases, we are led to suspect even more strongly that the Fibonacci pattern is a result of a quite general law of form, rather than directly a result of some strict requirement. So too for the property of projection in language, I would like to suggest. That is, certain

---

2 To preview: There is arguably a cost associated with making the growth pattern too complicated, such that a growth pattern like [α [β [X^0 γ]]] places a heavier burden on resources than does a format like [α [X^0 β]]. But the more complicated the pattern is allowed to be, the greater the reduction in c-command and containment totals. Thus, we expect language to settle on some 'minimax' compromise between the greater optimality of a more complicated growth rule, and the inherent costs of such further complication.

3 Of course, this invites the further question of whether those options are 'linguistically reasonable', or are ruled out for other reasons. I address this matter in section 5.

analyses propose that individual structures are not 'well-behaved' with respect to projection: small clauses, for example (see Moro 2000); see also the various proposals concerning exocentric or multi-headed structures (e.g., Williams 1994, Bouchard 1995, and Jackendoff 1977). If such analyses are on the right track, then it would seem misguided in principle to try to explain why projection is 'virtually conceptually necessary'.

1.2. Assumptions and Perspective

The results presented in this article are primarily mathematical in nature. This departs from the usual practice in linguistics of close and careful attention to the intricacies of natural language data, where a proposal is judged by its success in covering new empirical paradigms, or in reinterpreting recalcitrant patterns in more illuminating ways. The perspective taken here is highly abstract, several steps removed from detailed empirical descriptions and from highly ramified empirical predictions. Instead, the goal is to attempt to explore one kind of explanation for some broad empirical generalizations that seem more or less well-established. It will not be my purpose here to defend these empirical descriptions, nor to refine them or extend their coverage to new kinds of data. The predictions of this study, insofar as they can be construed as empirical at all, would be definitively falsified by a discovery that linguistic structures overwhelmingly tended toward some characteristic recursive shape other than a projective one, or had no such characteristic shape at all.[4]

I will assume that syntax consists of a computational system utilizing recursive Merge, which may apply both to items drawn from the lexicon and to the output of other Merge operations. I keep to the simpler case of External Merge throughout the article, setting aside the complications that arise in treating Internal Merge. I furthermore assume that Merge is subject to the Extension Condition, and limited to strict binarity.

An anonymous reviewer points out that binary branching may be one of the facts of language most in need of explanation in terms of efficient computation.[5] Accounts of "why language is that way" with respect to binarity

---

4 The matter is muddied by the observation that *any* binary branching structure can be decomposed into *some* combination of different 'projective forms' in the present sense, if all that matters is bare geometry. Nevertheless, the claims advanced here are not the merest triviality: The idea is that some particular projective structure is applied more or less consistently.

5 The same reviewer wonders whether the approach pursued in this contribution may shed some explanatory light on the matter. As explained below, under strict binary branching, c-command and containment totals are exactly equal. As treated here, this is simply a convenient accident, allowing both measures to be lumped together in a single measurement. The reviewer suggests that some principle of grammar may favor this sort of balance, or that perhaps this fact tells us something about which of the two relations is more important in language design. The second point seems promising at first: Completely flat structure minimizes containment relations absolutely, while maximizing c-command relations (though doing no worse than worst-case binary branching). Does this suggest that binarity is favored for c-command? Closer examination is not encouraging. For example, [[a b c] [d e f]], with a mix of binary and ternary branching, actually results in lower totals of *both* c-command and containment relations (20 and 14, respectively) than any strictly-binary arrangement of the same elements (22 of each, at best).

exist — for example, Kayne's (1984) notion of unambiguous path, his theory of antisymmetry (Kayne 1994), or the general notion that "what is necessary is also sufficient". Although the matter is in no way trivial, I simply adopt the usual assumptions in this regard, trusting that readers will find it at least familiar.

I also do not attempt to deal with the possibility that adjuncts may lie 'on another plane', as has sometimes been suggested, thus ruling out some interesting possibilities. Thus, the present approach can be seen as aligning with Kayne (1994) and Cinque (1999) in assuming that adjuncts are in fact specifiers with unexceptional geometry. If that assumption should prove incorrect, and adjuncts have some special status in terms of their branching geometry, then this study is leaving out another important case over and above Internal Merge.

1.3. A Preview of Comparing Recursive Patterns

As a first pass at the considerations to be explored here, suppose that a syntactic derivation has reached a stage where the following three objects remain to be combined:

(6) X^0, AP, BP

Let us take X^0 to be a bare lexical item, while AP and BP are internally complex objects constructed by Merge. For the purposes of this simplified example, let us ignore any distinction between AP and BP. The options for continuing the derivation are the following:

(7) [AP [X^0 BP]] (or [BP [X^0 AP]])

(8) [X^0 [AP BP]]

Is there any basis for choosing between (7) and (8) in terms of their effects on c-command and containment relations? There is. Let a be the number of nodes in AP, and let b be the number of nodes in BP. Since AP and BP are internally complex, $a, b > 2$. When two objects Merge, the number of new c-command relations defined is simply the sum of the number of nodes in each; likewise, the operation also creates the same number of new containment relations (as the new mother node contains all of the nodes in each). Thus, creating (7) defines $(b + 1) + (a + b + 2) = a + 2b + 3$ new c-command and containment relations. Creating (8), on the other hand, allows $(a + b) + (a + b + 2) = 2a + 2b + 2$ new c-command and containment relations, which is strictly greater. Thus, fewer such relations are (potentially) computed at this stage if the derivation 'grows' according to (7) rather than (8). As argued in more detail below, this gives us good reason for preferring (7) over (8) in terms of efficient computation, all else equal.

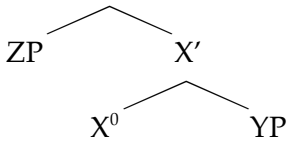
Needless to say, this departs from the usual way of thinking about these matters. For one thing, it is usually assumed that given some real example, only one of (7) or (8) could apply; the other choice would 'crash', failing to meet the requirements of the items involved. Moreover, only some of the c-command and containment relations defined would actually be exploited to carry real linguistic

relations. I return to these issues in more detail later on. For now, the idea is that if we find as an empirical matter that the configuration in (7) tends to predominate as a structural pattern, while configurations matching (8) are relatively rare, we might be able to explain that fact in terms of this kind of comparison.

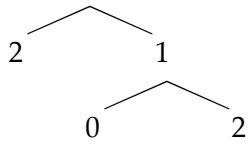
Note that (7) has the shape of an X-bar pattern of specifier, head, and complement, whereas (8) might correspond to a head taking a small clause complement, which seems to be a good deal less common (as an iterated pattern). What is at stake here has nothing to do with projection; questions such as whether X^0 is the 'head' of the construction do not enter into selecting one form over the other. Rather, the issue is one of branching form and its effects on c-command and containment relations.

In this light, consider the familiar X-bar schema in (9a). Setting aside the matter of projection (the fact that the complete syntactic object shares a lexical category label X with its head X^0), the relevant aspect for our purposes is that a complex syntactic object is formed by the particular recursive pattern in (9b).

(9) a.



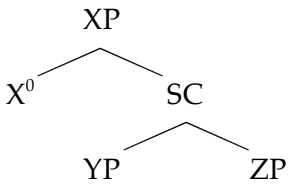
b.



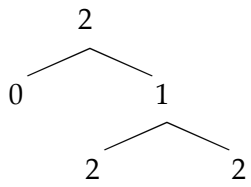
At first, it looks like (9b) is just a matter of 'bar-level' notation: 0, 1, and 2 correspond to X^0, X', and XP respectively. But there is a way of thinking about (9b) which does not require reference to explicit 'bar-level' features (a grammatical device that has been discarded from minimalist theory for good reasons). The objects in (9b) are merely a convenient notation for describing the particular recursive pattern embodied by the X-bar schema. That is, a 0 in (9b) is a terminal (a lexical atom), while 1 and 2 are defined recursively: A 1 is an object resulting from Merging a 0 and a 2, and a 2 is the result of Merging a 1 and a 2. This is a template for recursion, implicitly expandable 'all the way down'.

On the other hand, the option followed in (8) manifests a phrasal format distinct from the X-bar shape, as in (10). (10a) gives a familiar linguistic interpretation of the shape (a head taking a small clause complement, as in the analysis of the copula by Moro 2000). What is of interest for present purposes is the abstract recursive characterization of the shape in (10b).

(10) a.



b.

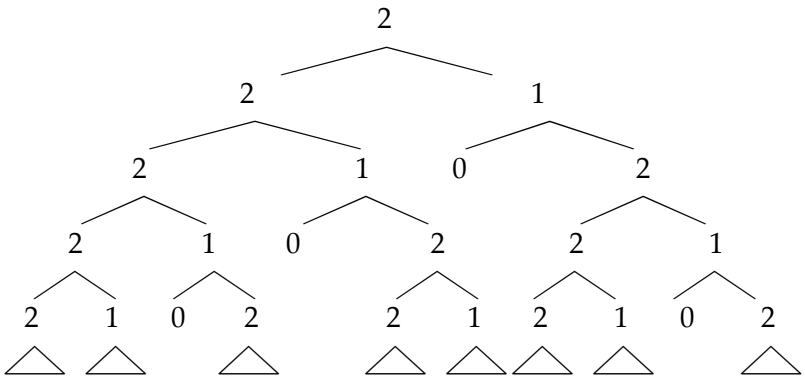


Lest this be misunderstood, let me hasten to point out that I am not claiming by the representation in (10b) that small clauses are X' categories, or anything of the sort. Instead, the point is that this structure can be characterized in terms of three kinds of geometric object. One is a terminal, X^0, labeled 0 in

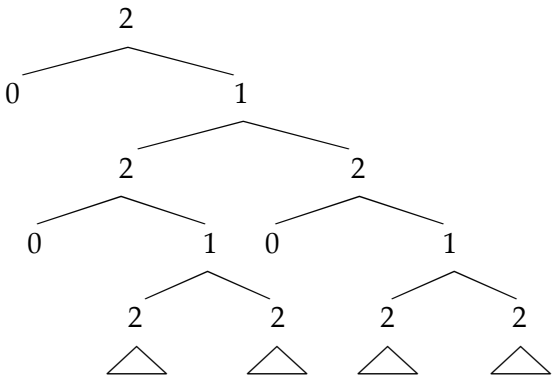
(10b). The other two objects (1 and 2) are distinguished by their recursive properties. The idea is that (10b) is an alternative to (9b) as a phrasal template. If this pattern continued, the nodes labeled 2 at the lowest level of (10b) would themselves be head+small clause structures of the same shape as (10b), potentially 'all the way down'. This would lead to different possible branching forms for linguistic structure.

I illustrate in (11) and (12) the results of recursively expanding the X-bar schema (9b) and the head+small clause pattern (10b). Expressions characterized by these patterns would fill some finite portion of these full branching spaces.

(11)



(12)



As is immediately clear, recursive expansion of the X-bar pattern creates a space of branching forms which is intuitively 'denser' than the space associated with the head+small clause pattern. This difference in 'branching density' turns out to be simply another aspect of the difference between (9b) and (10b), ultimately a part of the same fact underlying the local preference for (7) over (8). Put simply, the more densely the space of forms generated by a phrasal template branches, the better that phrasal template is for reducing the computational burden of c-command and containment relations. The relationship between recursive patterns (such as the X-bar format (9b) and the head+small clause format (10b)) and c-command and containment relations is the matter that will concern us in this article.

1.4. On 'Explaining' Projection

What is 'projection', exactly? This question was obscured by the notational

conventions of earlier theories, wherein the notion was almost trivial. Taking trees as real objects, 'projection' has to do with how non-terminal nodes are labeled; whichever daughter shares its categorial 'label' with the mother node has projected.

Within a minimalist theory such as Chomsky's (1995a) *Bare Phrase Structure*, this familiar notion suddenly becomes problematic. Chomsky proposes a set-theoretic interpretation of linguistic structure building. On that conception, it is no longer so straightforward to 'label' non-terminal 'nodes'. A device is stipulated to capture labels, but it seems somewhat *ad hoc*; Merge of α and β is taken to yield not the simplest object {α, β}, but rather {K, {α, β}}, K the label; this requires further complication in introducing the notion of 'Term', essentially so that syntactic operations 'skip over' the label as a potential syntactic object in its own right. Collins (2002) objects to this complication, pointing out that it goes "way beyond" what a minimalist theory of phrase structure requires.

More recent work seems largely to agree with Collins; 'labels' are now taken to be implicitly defined, with Merge keeping to the simpler, 'bare' output of {α, β}. Chomsky (2005) proposes that labels are identified by a search algorithm, and more recently has suggested that structures going beyond the head-complement format are 'unstable' in some sense (cf. Moro 2000 for a similar idea), and must be resolved by movement (Noam Chomsky, p.c.). Nevertheless, the idea of a 'label' perseveres, now motivated as a computational device carrying all information about a syntactic object relevant to further computation. Hornstein & Nunes (2008), following suggestions of Chametzky (2000) and Uriagereka (1998), challenge even this idea, arguing that for adjuncts at least, labels are unnecessary; the contribution of an adjunct to interpretation is understood via default 'conjunction' (here following Pietroski 2004).

Casting the matter in terms of interface interpretation in this way, we may well ask, with Wolfram Hinzen, whether forcing syntactic structure to reflect the relevant notions is really the right move:

> As for the notion 'head', why should phrase structure capture it, if the question of which of two lexical items that are merged becomes the head is decided by the *lexical* properties of these heads? (Hinzen 2006: 182)

Where does this leave us? It is hard to deny that there is something substantive to the notion of projection; a verb phrase, say, is different from a noun phrase, and this difference can be traced to the differences between verbs and nouns. But it is precisely the 'therapeutic' value of minimalism that it leads us to demand more than empirical justification for the postulation of various devices; the goal is not merely to discover what language is like, but to explain "why it is that way". Regardless of the descriptive value of projection, or even its 'usefulness' for interpretation or syntax-internal computation, there remains the problem of mechanisms. That is, what structural device or process actually underlies the phenomenon that surfaces as projection, and where does that come from? If the mechanism can be explained 'naturalistically' rather than teleologically (i.e., as emerging 'for free' rather than being motivated by its eventual function), then we are closer to the goal of truly 'Galilean' explanation of language.

1.5. Organization of the Article

This contribution attempts to cover some unfamiliar ground, exploring an unusual avenue of linguistic explanation at a highly abstract level. To avoid losing the way, it may be helpful to map out in advance where we are headed.

Section 2 examines one example of the kind of recursive pattern predicted by this account that is of particular interest: the specifier–head–complement configuration of X-bar theory. Here, I show that this pattern is fundamentally connected to the Fibonacci sequence. I include some speculation on the significance of this fact, and how it may relate to similar properties elsewhere in nature. Section 3 lays out the claim that c-command and containment relations are of central importance to certain aspects of linguistic computation, and that minimizing such relations results in more efficient computation. I briefly review several familiar empirical domains in which such concerns plausibly apply, and attempt to justify simply counting all such relations as an idealized measure of the relevant computational cost.

Section 4 tackles the problem of specifying what derivational patterns are available in principle to a Merge-based system. I develop a method to compare different patterns to each other in terms of c-command and containment relations, and map out how the various possibilities fare. The basic technique will be to compare different growth patterns to each other on the basis of the 'best trees' they can generate for a given number of terminal elements. Growth patterns will be partitioned into comparison sets on the basis of their complexity, and it will be shown that the best growth pattern from each comparison set is a member of the class of 'projective' patterns, with structural properties corresponding to endocentricity and 'non-head' maximality.

In section 5, I attempt to outline how the present study fits into the context of other current work, and where appropriate indicate why I have chosen to pursue an orthogonal line of inquiry. Section 6 concludes the article, drawing together the various threads and reviewing what has been established, and where it seems to point. Finally, I include an appendix presenting the formal results underpinning the claims made in section 4.

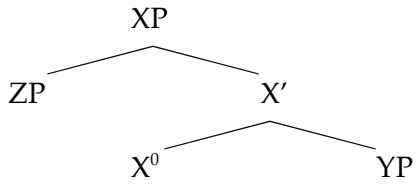
2. X-Bar Structure and the Fibonacci Sequence

In this section, I show that X-bar configurations are related in a fundamental way to the Fibonacci sequence.[6] Following Uriagereka's (1998) identification of Fibonacci patterns in syllable shapes and theme-rheme structure, this is of some biolinguistic interest in itself. The mathematical structure at issue is the specifier-head-complement configuration of X-bar theory in (13):

---

6 The Fibonacci numbers form the sequence 1, 1, 2, 3, 5, 8, 13, 21, 34,… defined recursively by $a_0=1$, $a_1=1$, and $a_n=a_{n-1}+a_{n-2}$. Named for Leonardo da Pisa (ca. 1200, also known as Fibonacci), the numbers were known long before to Indian thinkers. These numbers, and the related golden section, seem to be favored in the natural world in myriad ways, very few of which will be discussed here.

(13)



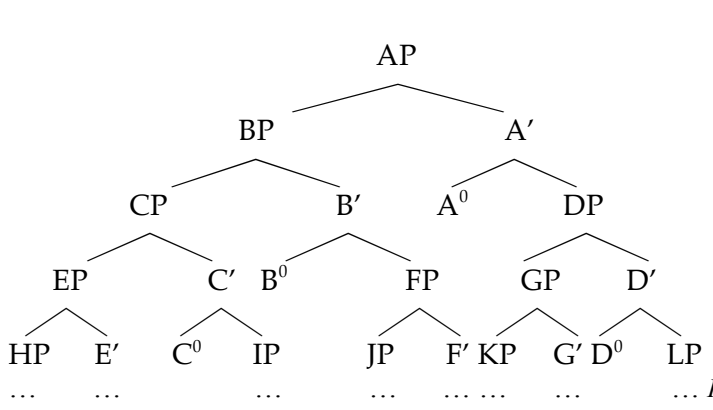
The object in (13) has played a central role in the empirical description of linguistic forms. The literature of X-bar theory is enormous; for some important developments, see Chomsky (1970), Jackendoff (1977), Stowell (1981), Kornai & Pullum (1990), Speas (1990), Kayne (1994), and Chametzky (1996). X-bar theory has been adopted widely even outside the Principles–and–Parameters tradition stemming from Chomsky (1981); see, for example, Bresnan (1982), Gazdar *et al.* (1985), and Pollard & Sag (1987, 1994). For now, suffice it to note that many researchers have taken (13) to be an important generalization about linguistic structure. Assuming so, we would like to know why phrases seem to pattern according to (13), if indeed they do; are there other possibilities? If so, why is (13) favored? As Hinzen puts it:

> What exactly does the X-bar scheme explain? And can its strictures be explained as following from more general and fundamental principles in the workings of the computational system? Or must we take it as an ultimate syntactic template that follows from nothing at all, accepting notions like headedness and projection as primitives? (Hinzen 2006: 180)

2.1. Iterated X-Bar: Fibonacci Numbers of Category Types

As noticed first by Carnie & Medeiros (2005), recursive expansion of the X-bar schema generates a Fibonacci sequence of bar-level categories at successive levels of embedding. Let us take the X-bar schema as recursively defining an X-bar space, and imagine 'filling' this space, such that all possible specifiers and complements are realized, each with their own specifiers and complements, 'all the way down'. If the X-bar schema is iteratively expanded in this way, the number of XPs, X's, and X^0s at successive levels of depth in the structure each form a Fibonacci sequence. This can be seen in the partially expanded structure in (14).

(14)



XP	*X′*	*X⁰*
1	*0*	*0*
1	*1*	*0*
2	*1*	*1*
3	*2*	*1*
5	*3*	*2*
… *Fib(*n*)*	*Fib(*n*–1)*	*Fib(*n*–2)*

Recall that Fib(n) is defined recursively by a_0=1, a_1=1, and a_n=a_{n-1}+a_{n-2}.[7] In the X-bar schema, each XP at depth n introduces another XP at depth n+1 (its specifier), and another at depth n+2 (its complement). Thus, the number of XPs at depth n is the sum of the number of XPs at depth n–1 and n–2. There is a single XP at level 0 (the root node), and one at depth 1 (its specifier). Thus, letting XP(n) represent the number of XPs possible at depth n, XP(n) = Fib(n). Each X' at depth n is introduced by an XP at depth n–1, so the number of X's at depth n, or X'(n), is Fib(n–1). Finally, each X^0 is introduced by an XP at depth n–2, so X^0(n) (the number of X^0s at depth n) is Fib(n–2). As a further consequence, the sum of number of objects of all types at each level of depth (i.e. XP(n) + X'(n) + X^0(n)) is a double of a Fibonacci number (2*Fib(n)) everywhere except at the root.

Figure 1 below provides a more perspicuous way to visualize how the Fibonacci sequence arises in the fractal space of forms generated by the X-bar pattern. Here, linear order is mapped to the counter-clockwise direction around the circle, starting at the top/'north' (assuming specifier–head–complement order). The binary Merge at the root of the tree corresponds to a division of the circle exactly in half; further binary branching deeper in the tree divides the relevant portion of the circle in half again. Where terminals occur in the expanded X-bar schema, the relevant portion of the circle is blacked out (no further subdivision will occur there). For example, in an X-bar tree the first terminal down from the root (the head of the root XP) occupies the left half of the right branch. Thus, the quarter circle between south and east is colored in. The next head down from the root is the head of the specifier phrase, corresponding to the shading of the eighth of the circle between the southwest and west directions. This process continues indefinitely; the result is a fractal diagram with Fibonacci numbers of successively smaller fractions blacked out, illustrating how the space of possible binary-branching forms may be 'populated' by terminals under perfect (infinite) iteration of the X-bar pattern of recursion.

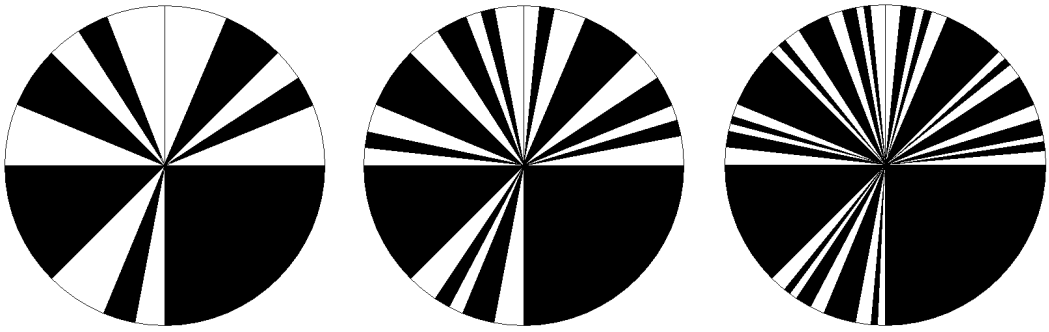


Figure 1. Three steps in the recursive expansion of X-bar space

[7] Frequently, the Fibonacci sequence is defined with a_0=0, a_1=1. It should be clear that the choice of index at issue is arbitrary, and irrelevant to the point being made here.

2.2. Fibonacci String Lengths and X-Bar Analyses

There is another sense in which X-bar structure is related to the Fibonacci sequence. This fact is related to the question of what X-bar analyses can be assigned to a linguistic string of a given length. By an X-bar analysis, I mean an assignment of bracketing such every phrase contains a head, and up to two other phrases in the usual configuration of specifier and complement. That is, expanding the X-bar schema top-down, each phrase XP may have any of the following shapes, but no other possibilities are allowed:

(15) a. $XP = X^0$
b. $XP = [\ X^0\ YP\ ]$
c. $XP = [\ ZP\ [\ X^0\ YP\ ]]$

Put another way, the X-bar scheme is taken to be a 'ceiling', but not a 'floor', on the internal complexity of a phrase. Assuming so, a number of different X-bar analyses are available for any string. Let us call the 'depth' of an analysis the maximum level of embedding of any element in the tree it assigns to the string. For example, a string of length 1 must have depth 0 (i.e., it is a trivial tree consisting of a single node), string length 2 requires depth 1, and string length 3 requires depth 2. For greater string lengths, some analyses will have different depths than others. As a function of the string length, we can identify the maximum depth of any possible analysis (clearly, 1 less than the string length), and also the minimum possible depth.

Fibonacci string lengths are *minimal depth milestones*, in the sense that a string of length Fib(n) is the first string length with a greater minimal depth than the previous string length. That is, a string of length 4 has a minimum depth of 2, the same as the minimal depth of string length 3; 5 is the first string length which forces an analysis of depth 3; likewise, string length 8 is the first with a minimal depth of 4, and so on.[8] Of course, real strings may have deeper analyses than the minimum. The point is simply that Fibonacci numbers have significance in terms of best-possible analyses, since minimal depth analyses are the 'best trees' within X-bar for a given number of elements, in terms of minimizing c-command and containment relations (see section 4 and the Appendix for discussion). As an illustration, consider (16) below:

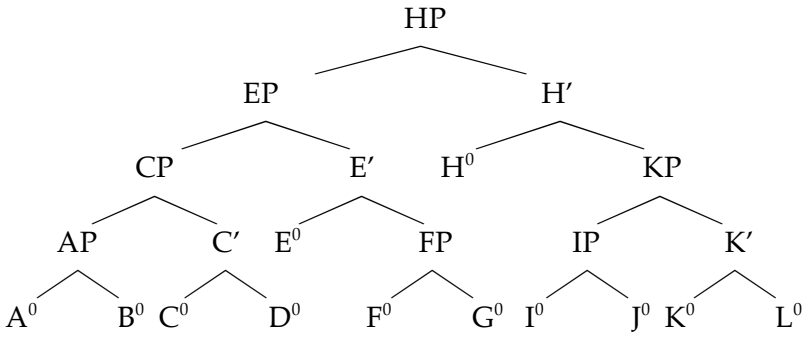
---

8 This follows directly from the observation in the section 2.1 and this well-known identity:

(i) $\sum_{i=o}^{n} \text{Fib}(i) + 1 = \text{Fib}(n+2)$ (This identity is easily proven by induction.)

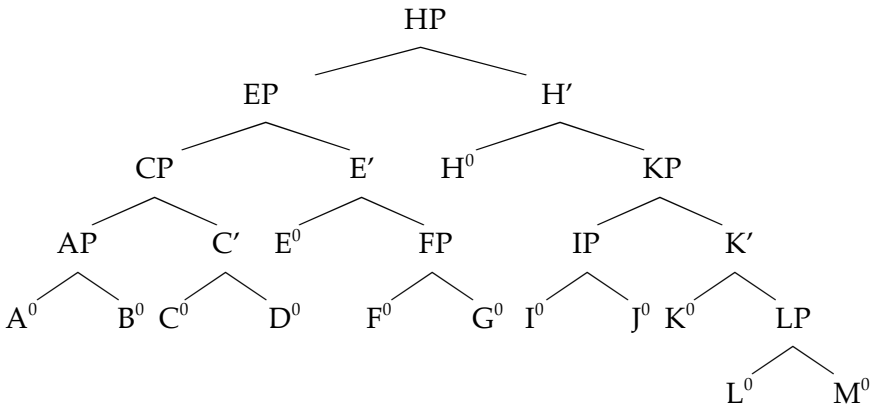
To see why, consider the X-bar analysis that packs the longest string possible into an X-bar analysis of a given depth. Given (23), in this analysis, all of the categories (including XP and X') at the greatest depth n are formatives in the surface string; thus, Fib(n) XPs + Fib(n–1) X's + Fib(n–2) X^0s, plus all of the X^0s introduced at lesser depths: Fib(n–3) + Fib(n–4)… + Fib(0). Adding one more terminal to the string forces the tree to depth n+1.

(16)



The representation in (16) contains as many terminal nodes as possible for a depth 4 tree (*viz*. 12). The next string length, 13, is a Fibonacci number, and it is the first string length which forces the X-bar analysis to a minimal depth of 5. That is, to add another terminal element to (16) while adhering to the restrictions imposed by the X-bar format (understood as in (15)), one of the nodes at the bottom-most layer of the tree must be expanded, bringing the depth of the tree to 5 for the first time. (17) is an example of such a 'milestone' tree; no rearrangement of this number of elements into a structure consistent with the X-bar pattern has less depth.

(17)



2.3. Are the Fibonacci Properties of X-Bar Significant?

It is tempting to see the appearance of the Fibonacci sequence in the X-bar pattern as being deeply significant in itself. But the X-bar schema is after all a very simple mathematical object, and there may be nothing particularly magical about the appearance of the Fibonacci sequence in the structures it generates. Their appearance in this domain could be no more of a surprise than their appearance in the family trees of bees, or in Fibonacci's idealized rabbit populations, or in the number of metrical possibilities for a line of Sanskrit poetry, or any of the myriad situations these numbers describe. To put it another way, it could be that these properties are an accident of no 'real' significance, or worse, merely a reflection of mathematical simplicity in linguists' description of language, rather than a property of language itself.

Yet it is undeniable that patterns related to the Fibonacci sequence play an important role in nature, especially in optimal packing and optimal arboration.

For example, botanical elements emerging from a central growth point tend to spontaneously organize into Fibonacci numbers of spirals, winding in opposite directions. In that case, it is known that the pattern is indeed the 'best possible' (dynamic) solution. Likewise, the pattern shows up in the branching patterns of many plants (e.g., sneezewort), and in a different sense in the proportions governing asymmetric branching in mammalian bronchial structure. The list goes on; see Uriagereka (1998) for discussion and further examples, including other Fibonacci patterns in linguistic structure. It seems that the pattern plays a 'spooky' role in nature (particularly in situations related to optimal self-similar growth). Thus, finding such a pattern in phrase structure suggests that this may be another manifestation of 'laws of form', reinforcing the biolinguistic suspicion that something deeper than just biology or linguistic principles are at work; the property may well "follow from principles of neural organization even more deeply rooted in physical law" (Chomsky 1965: 59).

All of this is intriguing, but of course it remains to specify exactly in what sense the X-bar pattern is optimal. This article attempts to go some distance towards exploring the details, but in the end falls short of motivating the X-bar pattern alone. Nevertheless, the weaker but more general conclusion reached below seems promising, namely that branching forms which look like a version of the X-bar pattern generalized to any number of specifiers are optimal. For now, I would like to point out the following intriguing analogy with plant growth.

2.4. An Analogy with Idealized Plant Growth

Notice that in a binary-branching tree, each node is c-commanded and contained by a number of nodes equal to its depth in the tree. For reasons clarified in the next section, I will propose that the number of c-command and containment relations in a syntactic tree indicate a computational cost. This cost can be intuitively pictured as a 'force' pulling toward the root of the tree, in the sense that the deeper in the tree a given piece of structure is, the greater the number of c-command and containment relations it incurs.

Then the problem faced by the syntactic system is analogous to the following idealized problem of plant growth. Suppose that a plant is 'binary branching', and at each branching point, either new structure can become a terminal leaf, gathering sunlight but preventing further growth, or can grow a non-terminal stem which divides again in two. The plant 'desires' to grow as many leaves as possible (to gather sunlight energy more effectively) without making the resulting structure too tall/spindly. Vertical growth magnifies the structural strain involved in supporting the structure against gravity, wind, and so on, which is increasingly severe for each additional increment of growth away from the root (a longer stalk serving as a more effective lever for a given wind strength, and so on). Here, nature is searching for some compromise between growing as many leaves as possible, and not making the resulting form too tall.

If plant growth places two leaves at the very first branching, it is done growing. If it places one leaf at every branching point, only one branch will then remain available for further growth, resulting in a final form with a single stalk

(in linguistic terms, it is unidirectionally branching). Delaying leaf-generation for some number of branchings yields better results over the long term, as the final form will be bushier, shorter, and less likely to topple over from wind or its own weight. The very best final form would branch everywhere until spontaneously producing only leaves at the last generation. Of course, plants *grow*, making the notion of 'last generation' unavailable. What seems desirable is to strike some mini-max balance between growing as many leaves as possible immediately, and investing in optimality for future growth by growing more branches.

I will propose that syntax faces an equivalent problem (physical interpretation of the details aside, of course). That is, the 'cost' of branching structure grows with depth, such that each increment of deeper branching is costlier than the last (inducing more potential c-command and containment relations). The local 'force' on terminals is reduced by packing them as close to the root as possible, which is antagonistic to global optimality (each terminal which is too close to the root 'closes off' options for other structure, which must instead appear even deeper in the tree). In both botany and syntax, the Fibonacci pattern is a good compromise to this problem; perhaps even the best, depending on further details of the system.

3. C-Command and Containment in Linguistic Computation

The primary tool of investigation in this article is the comparison of hierarchical structures on the basis of the number of c-command and containment relations they encode. Such relations are central to linguistic computations of various sorts (e.g., long-distance dependencies). Given the recent focus on principles of efficient computation, the hypothesis is that the derivation of structures with fewer such relations represents less of a computational burden. Insofar as different derivational patterns lead to structures with differing numbers of c-command and containment relations, there is then a basis in computational efficiency for preferring some derivational patterns over others. If we find that the recursive patterns which seem to characterize natural language are drawn from the patterns which are optimal in this sense, we may suspect that this aspect of phrase structure has a minimalist explanation.

I adopt the familiar definition of *c-command*, as follows:

(18) *C-Command* (Reinhart 1976: 32)

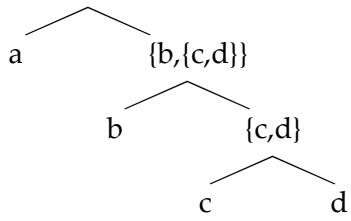
Node A c-commands node B if neither A nor B dominates the other and the first branching node which dominates A dominates B.

We will also be interested in *containment* (i.e. irreflexive domination), taken as the transitive closure of the 'immediately contains' relation. Note, first, that the totals of these relations are always equal in binary-branching trees. For each node α in a tree, the number of nodes which contain it is equal to its depth in the tree. Since the tree is binary-branching, the number of nodes that c-command α is also equal to its depth, because each node which contains α immediately contains a node β not containing α, which thus c-commands α; no other nodes c-command α.

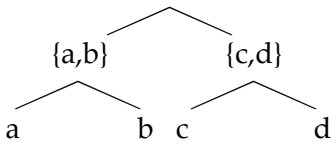
3.1. A Simple Observation

The point of departure for the present contribution is the simple observation that different patterns in Merge result in different totals of c-command and containment relations, even for the same input (number of terminals). For a simple example of this, consider sets (19) and (20).

(19) {a,{b,{c,d}}}



(20) {{a,b},{c,d}}



These structures have equal numbers of terminals and of non-terminals, yet (19) has more c-command relations (12, compared to 10 in (20)). This is shown in (21) and (22), a listing of all the c-command relations present in (19) and (20), respectively (read "x: y, z, w" as "x c-commands y, z, and w").

(21)

{a,{b,{c, d}}}:	–
a:	{b,{c,d}},b,{c,d},c,d
{b,{c,d}}:	a
b:	{c,d},c,d
{c,d}:	b
c:	d
d:	c

$\Sigma = 12$

(22)

{{a,b}{c,d}}:	-–
{a,b}:	{c,d},c,d
{c,d}:	{a,b},a,b
a:	b
b:	a
c:	d
d:	c

$\Sigma = 10$

For completeness, I list all containment relations for (19) and (20) in (23) and (24), respectively. Here, "x: y, z, w" means "x contains y, z, and w".

(23)

{a,{b,{c,d}}}:	a,{b,{c,d}},b,{c,d},c,d
{b,{c,d}}:	b, {c,d}, c, d
{c,d}:	c, d

$\Sigma = 12$

(24)

{{a,b}{c,d}}:	{a,b},a,b,{c,d},c,d
{a,b}:	a,b
{c,d}:	c,d

$\Sigma = 10$

3.2. C-Command in Linguistic Relations

The notion of c-command is central to numerous linguistic relations. Reinhart's (1976) original concern was describing the distribution of anaphora. While still relevant to binding theory, c-command is also implicated in linearization (Kayne 1994), the determination of relative scope (May 1985), and the probe-goal mechanism of Chomsky (2000, 2001), the latter taken to underlie long-distance agreement and to be a pre-condition for displacement.

Epstein *et al*. (1998) provide a natural reason for the ubiquity of the c-

command relation in terms of a derivational view of syntax. As they point out, c-command amounts to the condition that syntactic objects can enter into linguistic relations with elements of the sub-tree they are merged with. This suggests a view of c-command as following from a search operation (potentially) accompanying each Merge operation. The property of Minimality, as encoded by principles such as the Minimal Link Condition, Shortest Move, Attract Closest, and Relativized Minimality (the relevant literature is vast; see Chomsky 1995b, Rizzi 1990, among many others), reinforces this interpretation of c-command. The basic observation is that in configurations like (25), where X could enter into a dependency with either Y or Z but Y is 'closer' to X in some appropriate sense than Z is, a dependency may hold between X and Y but not between X and Z.

(25) X … Y … Z

This closeness is usually measured by c-command relations: If Y asymmetrically c-commands Z, then Y is closer to a c-commanding X than Z is. To a first order of approximation, we might reasonably say that syntax seems to 'minimize links', presumably for reasons related to efficient computation. The idea is that long-distance dependencies reflect a search operation 'probing' for a 'goal' in the searched category in a top-down fashion (Chomsky 2001). Once an appropriate goal is found, the search terminates, thereby blocking a dependency with a more deeply embedded but otherwise legitimate goal (so-called 'intervention effects', possibly unifiable with the A-over-A Condition).

As one aspect of 'least effort' conditions on efficient computation, Chomsky (2000: 99) explicitly includes principles aimed to "reduce 'search space' for computation: 'Shortest Movement/Attract', successive-cyclic movement (Relativized Minimality, Subjacency), restriction of search to c-command or minimal domains, and so on". The last point is especially significant for our purposes: In terms of individual instances of search, the burden is less if a smaller domain is searched. But note that the total number of c-command relations in a syntactic object is simply the sum over the size of domains that have (potentially) been searched during its derivation. Thus, it is a natural extension of the drive to restrict the domains for individual searches to prefer structural patterns leading to lower c-command totals, since that amounts to restricting the aggregate domain for iterated searches.

C-command totals may be taken to indicate computational cost in other ways as well. Beyond the search interpretation of the probe-goal mechanism, Kayne's (1994) Linear Correspondence Axiom can be understood as a process 'reading' c-command relations and deriving linear order. Moro's (2000) theory of dynamic antisymmetry reinforces this view of linearization as a computational process at the interface, for him crucially applying after syntactic displacement has resolved points of symmetry. Likewise, scope is affected by displacement, again suggesting that some process 'reads' c-command relations at the interpretive interface. It seems natural to suppose that processes of linearization and the determination of scope are less burdensome if applied to objects with fewer c-command relations.

3.3. Containment Computations

There are reasons to believe that certain linguistic computations are 'measured out' by containment relations, such that minimizing the number of such relations improves computational efficiency. For example, Chomsky & Halle (1968) note a relationship between stress levels and hierarchical set structure in complex expressions. They propose a cyclic rule of stress assignment (the Nuclear Stress Rule) that re-computes stress in successive applications from most-to-least embedded levels. See Halle & Vergnaud (1987) for a broadly similar system, as well as Hayes (1995). In all of these theories, the stress on individual items may potentially be readjusted at each level of embedding. Importantly, an arrangement like (20) involves fewer total (potential) adjustments of the stress levels on individual elements than (19). That is, in (19) the most deeply embedded elements (*c* and *d*) will be subjected to three cycles of stress computation; the element *b* will undergo two cycles, and *a* just one: The total is 9 (potential) readjustments of individual stress levels. On the other hand, in (20) each element is twice-embedded, hence subject to 2 cycles, for a total of 8 potential readjustments. This suggests that assigning stress to (20) is a simpler computation than doing the same for (19).

Along the same lines, consider theories that relate displaced elements to their position of canonical interpretation in the way that Head-Driven Phrase Structure Grammar (Pollard & Sag 1987, 1994) does.[9] To encode discontinuous dependencies (e.g., in *wh*-questions), HPSG utilizes a feature on a verb (a SLASH feature in the HPSG parlance) marking its semantic deficiency, which propagates up the tree along the path of dominating nodes until it encounters a category that can satisfy it. If some such mechanism underlies displacement phenomena in general, then one natural condition of efficient computation is that the feature propagation path should be as short as possible. Maximally balanced trees like (20) provide a scaffolding with the minimal propagation path-length sum possible; in general terms, the 'average' containment path is shorter in such a tree, and the worst-case paths are shorter than in any other structure.

3.4. Is Counting Enough?

I will resort to simply counting all of the c-command (equivalently, containment) relations in a structure, adopting the working hypothesis that this is a reasonable proxy for the 'real' computational cost incurred in actual expressions. It might be objected that counting all c-command relations may overestimate the relevant cost in important ways. For one thing, it is often assumed that in a given configuration, the relations for which c-command matters are one-sided. Thus, when α and β merge, only one (say, α) can search the other; dependencies cannot be established from β into the interior of α. Moreover, the very fact of intervention means that not all probe-goal searches are computed; the search

[9] Of course, HPSG is a model-theoretic approach to syntax. It is not clear that concerns of 'efficient computation' are as relevant to such an approach as to proof-theoretic derivational accounts pursued within the Principles–and–Parameters tradition.

stops once the first legitimate target is found. Thus, it seems we are crucially over-counting the c-command relations that should matter for such a comparison. Similar concerns apply to containment; for the case of stress assignment, it seems only the containment relations involving terminals matter for optimality.

This is a necessary casualty of the idealizations here. To restrict dependencies such that only one of the operands of Merge may search the other requires some basis for the asymmetry. Given the range of constructional options considered here, there simply is no way to reconstruct such an asymmetry on configurational grounds in full generality. Since any other grounds for the asymmetry (say, properties of the individual lexical items involved) are ignored as well, we shall have to live with this. Similarly with the intervention effect: Without knowing what dependencies might actually be established or not, we are left with a bare scaffolding of possible dependencies and no way of choosing how it might be filled out. The only basis for comparison is the scaffolding itself.

Even so, I think the approach here is not unreasonable. Recall that the goal is to find a basis for selecting certain structural patterns over others. At this level of idealization, it may make sense to abstract away from the details and consider the total space of possible relations latent in branching forms themselves. However the possibilities are eventually exploited in particular expressions by some defined relations entering into linguistic computations while others do not, it is a fact that some structures put a tighter cap than others on the computational cost that could be incurred in principle.

Furthermore, it is a crucial point that the measure of computational cost need not be strictly accurate for our purposes; all that is important is that it reflects the *relative* optimality of the structural options being compared. In this regard, it is encouraging to note a general property of scale-invariance in the comparison between different recursive possibilities. As we will see, the self-similarity of the patterns to be explored implies that if one pattern produces fewer c-command and containment relations than another in small domains, their relative optimality will not be reversed in larger domains. The same property of self-similarity suggests that the comparison will tend to go the same way if domains are restricted in principle in equivalent ways.

The hypothesis — and it is only that — is that at this level of abstraction, this simple expedient of counting will suffice to illuminate at least the outlines of where such an approach will lead. But suppose it turns out that simply counting total numbers of c-command and containment relations is wrong in some fundamental way, and a more detailed look at the properties involved leads to different measurements, making different predictions. Even so, I would like to suggest that such predictions should be taken seriously, and their explanatory potential explored. In other words, the methodology employed here may prove to be too simplistic, but I think the underlying concerns deserve attention, in that (to my knowledge) computational efficiency in this form has not been examined before, and the potential for 'deep' linguistic explanation in these terms appears promising.

4. Optimal Syntactic Growth Modes

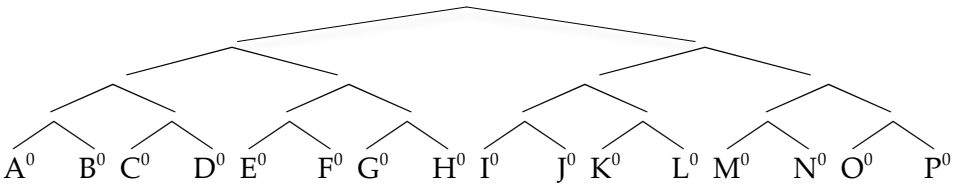
What I propose to investigate and compare below are phrase structural patterns, in the sense of characteristic aspects in the branching geometry formed by Merge applying recursively to lexical items and its own output. The hypothesis being entertained is that the forces which govern the process, in the sense of selecting some binary-branching structures over others, will give rise to identifiable and repeated tendencies (what might be thought of as 'optimal growth modes'). To determine what tendencies we might expect, I generate all possible patterns that could be used as consistent 'phrasal templates' to build infinitely recursive structures from lexical atoms, and develop a technique to compare them to each other.

4.1. A Domain for Terminals

One condition that will need to be imposed is that the recursive templates include a characteristic place for terminal elements. This makes a good deal of sense on several levels. First, the objects are recursively defined, which requires some 'base step'; it is hard to see what aspects of branching structure could provide this other than terminals. From another point of view, these are ultimately discrete, finite patterns, built bottom up from lexical items; they are 'about' structuring terminals into larger structure. Without terminals to 'ground' the patterns, there can be no distinctive shape, hence no 'pattern' at all; the only rule would then be 'anything goes'.

The concern in this regard is structures like (26) below, which are 'maximally balanced', with all terminals at the same depth (or at two adjacent levels of depth. These structures provide absolute minimization of c-command and containment relations.

(26)



If the concerns in this article really do 'matter' in the determination of structure, why do we not see such forms in natural language? If the only problem were optimizing *at once* the positioning of a full set of elements, we would indeed expect to see something like this.

But one guiding theme in minimalist work is the idea that syntactic forms are to be explained dynamically, by local (informationally limited) optimization at each step of a syntactic derivation. In these terms, the structure above looks decidedly unnatural. To actually derive such a form, Merge must apply as symmetrically as possible. This involves unbounded 'vertical' information flow at each step; the internal structure of syntactic objects must be accessible 'all the way down' so as to match objects (terminals, pairs of terminals, pairs of pairs of terminals, etc.) appropriately. But even this 'local' (i.e. one Merge operation at a

time) matching of object structures is not enough. The derivation must be kept in appropriate synchrony across the entire set of parallel sub-derivations; if one process of merging terminals into ever-larger sets proceeds too many steps beyond other combinations occurring in parallel, we may be left with a final stage where only unmatched objects remain. Information must thus be shared 'horizontally' as well, in effect amounting to global pre-planning of the derivation.

We can identify a parallel situation in botanical growth. Recall the idealized problem of leaf-placement in section 2.4: The ideal representation for solving the problem produces no leaves until the last generation, when only leaves are produced. There is something distinctly unnatural about this; organic growth proceeds by a local logic, where notions such as 'final form' have no power to shape the dynamics of growth. Similar concerns apply to the pattern of Fibonacci spirals in phyllotaxis: If the only problem were to pack at once a certain number of elements into a limited space, a hexagonal lattice structure would be best. But the observed patterns *grow*, with the result that what we in fact observe is not the best form, but the best growth pattern, a crucial distinction.

Given the dynamic view of syntax adopted here, similar constraints are expected to apply: The best configuration is 'ungrowable'. Parallel to the phyllotactic case, we expect to observe at best an optimal derivation, not an optimal final representation, because the dynamic system is limited by a fundamental locality. This is why (26) is not predicted here; no local pattern of growth can produce it.

4.2. Possible Growth Modes

Such concerns lead us to expect that the considerations which enter into derivational choices will be limited by an informational horizon. Recall that one of the problems with (26) was that it required syntactic objects to be matched 'all the way down'. Limiting this informational flow means that only some of the recursive structure of the operands of Merge is 'visible' to optimization concerns. For example, if one level of internal structure can be examined, then terminals can be distinguished from more complex objects. Allowing two layers of structure to be visible allows further distinctions, which allows more internal complexity in recursive patterns, and so on.

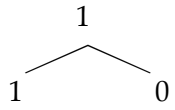
As an idealization to aid the investigation of these matters, I will suppose that whatever pattern might be found will be consistent (i.e. deterministic). A consistent recursive scheme carried out within a finite derivational window can be described by a finite number of distinct 'types' of syntactic object (terminals, or objects recursively defined as the result of Merging other terminals or recursively defined objects), which 'loop' into each other in a finite cycle.

4.2.1. Notational Conventions

To allow the full range of recursive possibilities, let us simply use the natural numbers to represent the relevant distinctions among outputs of different Merge operations, reserving 0 for terminal elements. Let us furthermore use the largest

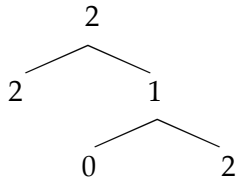
number in a pattern to designate the root symbol (held constant, under the 'top-down' formulation discussed below). Here, we will take the appearance of the same number on two different nodes to mean that the structures so labeled have isomorphic recursive structure. In these terms, the simplest recursive pattern (both including terminals and allowing indefinite recursion) will be represented as below:

(27)



Likewise, in this formulation the X-bar specifier-head-complement pattern will have 0-level terminals marked as 0s, while 'single-bar-level' intermediate categories are 1s, and 'phrases' are 2s.

(28)



Thus, the numerical designations might be thought of as something like a generalization of conventional 'bar-level' notation. To be clear, this is *not* a proposal about reviving bar-level notation as an explicit grammatical device, thus violating Inclusiveness. Instead, the notation is a device for reasoning about possible derivational sequences; the relevant information is not to be understood as somehow reified in any way 'on' the node, but is a matter of information that is in the way the derivation itself proceeds. If these patterns do characterize natural language, that fact presumably emerges from dynamic considerations, rather than being explicitly enforced by some mechanism like 'bar-level features'.

Insofar as a pattern is consistent, its elements (other than 0) can be characterized by what amount to 'rewrite rules' (again, this is a matter of investigational convenience, not a proposal for a 'real' grammatical device):

(29) i → j k i in {1, 2,… *n*}; j, k in {0, 1, 2,… *n*}

The simplest structure (27) can thus be expressed as in (30), and the X-bar schema as in (31):

(30) 1 → 1 0

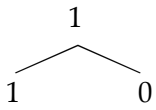
(31) 2 → 2 1
1 → 2 0

4.2.2. Generating All Possibilities

Let us now set to exploring the options systematically. If the 'derivational

window' is as small as possible (i.e. the growth pattern is as simple as possible), then there is only one option for how to build recursive structure from terminals. I call this the 'spine', for obvious visual reasons (intuitively, it generates a uni-directionally branching tree); I will likewise use descriptive names for the other patterns for mnemonic convenience.

(32) 1 → 1 0 ('spine'):



We obviously need at least this much structure to have recursion at all. Ignoring linear order (as I do throughout), and requiring the pattern to be built recursively from terminal elements and the output of Merge, for distinct objects 0, 1 the other combinations can be ruled out (1 → 1 1 is not built from terminals, while 1 → 0 0 does not recurse).

Moving on to the next level of complexity in sequencing Merge, we consider patterns involving two types of non-terminals (equivalently, two-stage sequencing of Merge operations). Given the remarks above, we have at first pass $6^2 = 36$ distinct options for recursive patterns involving two order-irrelevant Merge rules (i.e. non-terminal characterizations) defined over three object types (0, 1, 2); for arbitrary n, there are $(n(n+1)/2)^{n-1}$ options. Being a little more careful, we can restrict this further by ruling out the following types of characterizations:

(33) i → i i does not terminate (DNT)
n → 0 0 does not recurse (DNR)
n → n 0 isomorphic to the Spine

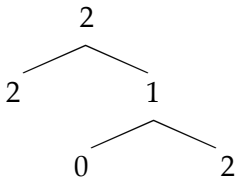
That is, any object which immediately contains two isomorphic copies of itself cannot be recursively constructed from terminals. If the root node (designated as the largest number n) consists of two terminals, recursion is impossible. Finally, if the root node consists of a terminal and an object isomorphic to the root, it is isomorphic to the spine (1 → 0 1), hence is not really a member of the higher-order comparison set. The table below lists all the options for the comparison set built from {0, 1, 2}; non-viable options are grayed out.

	2 → 2 1	2 → 1 1	2 → 1 0
1 → 2 2	DNT	DNT	high-headed D-bar
1 → 2 1	DNT	DNT	high-headed X-bar
1 → 2 0	X-bar	D-bar	(spine)
1 → 1 0	spine of spines	pair of spines	(spine)
1 → 0 0	double-headed spine	DNR	DNR

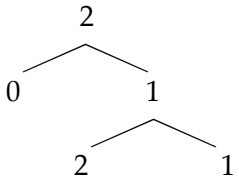
Table 1: Options for the comparison set built from {0, 1, 2}

I have also grayed out the option described as a 'pair of spines', which, as the name is intended to suggest, consists of two spines merged at the root. It should be clear that this is not a repeating structure; the configuration at the root is unique, and thus it is not a growth pattern in the desired (basically, self-similar) sense. I illustrate the remaining options below, including their repeating 'molecular' structure as a partial tree diagram.

(34) a. 2 → 2 1 ('X-bar')
1 → 0 2

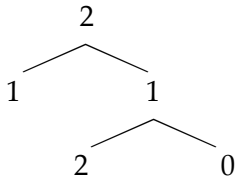
b.


(35) a. 2 → 1 0 ('high-headed X-bar')
1 → 2 1

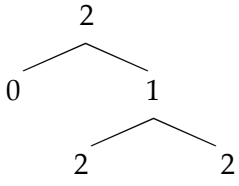
b.


Options (34) and (35) form a natural pair, as do (36) and (37) below, in that the members of the pairs are really the *same* recursive cycle caught at different times, with a different selection of which non-terminal serves as the root. I call the member of each pair of patterns in which the terminal occurs nearer to the root 'high-headed'. See the discussion in 4.3.3.2 below.

(36) a. 2 → 1 1 ('D-bar')
1 → 2 0

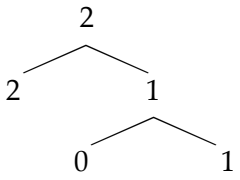
b.


(37) a. 2 → 1 0 ('high-headed D-bar')
1 → 2 2

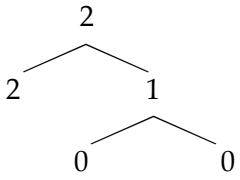
b.


This pair (again, really different 'snapshots' of the same pattern) has a fundamental symmetry; the D in D-bar is meant to stand for 'double' for this reason.

(38) a. 2 → 2 1 ('spine of spines')
1 → 1 0

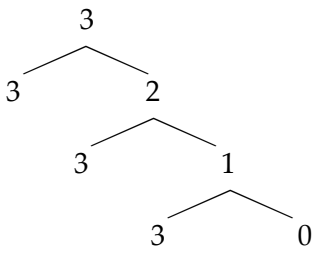
b.


(39) a. 2 → 2 1 ('double-headed spine')
1 → 0 0

b.


Enumerating all of the options for further comparison sets (allowing three stage Merge sequences/three non-terminal types) would be a good deal more tedious. For illustrative purposes, I include just one of the options. This represents the 'projective' geometrical format, and thus is the optimal member of its class (for reasons discussed below, and proven in full generality in the Appendix). Intuitively, it corresponds to the structures described by Jackendoff's (1977) 'uniform three-level hypothesis', an X-bar-like structure with two specifiers. In other words, it is a version of the X-bar schema utilizing three non-terminal types; hence, '3-bar'.

(40) a. 3 → 3 2 ('3-bar')
2 → 3 1
1 → 3 0

b.


4.3. Comparing Growth Modes

Now that we have developed a way of enumerating the possibilities for recursive growth modes, we turn to the task of comparing them to each other. Recall the fundamental observation underlying this investigation, that building structure in some ways results in fewer c-command and containment relations than other options. I have argued that having fewer such relations lessens the computational

burden for the derivation. The hypothesis is that this results in a preference for patterns in the application of Merge that will tend to reduce c-command and containment relations. Our goal in this section will be to develop a technique to compare the recursive options we have enumerated on the basis of their consequences for c-command and containment totals.

4.3.1. Comparison Sets Based on Cycle Complexity

Each of the recursive patterns we are considering is defined within the bounds of some fixed amount of sequential complexity. Some patterns have more or less internal structure than others: The spine is 'simpler' than the X-bar schema. The X-bar schema requires more in the way of (relatively local) information flow to structure the derivation appropriately. Different choices of the size of the derivational window (i.e. the number of different types of object, or equivalently, the number of derivational steps in a characteristic cycle) will partition the possibilities into natural comparison sets. That is, we will compare recursive patterns of comparable complexity to each other. In present terms, we will be comparing patterns that can be specified with the same number of symbols, so that a comparison set will consist of all the recursive possibilities that can be described with numbers from 0 to some fixed *n*.

4.3.2. Direct Comparison

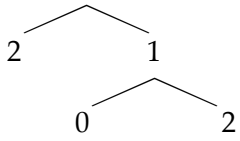
How can one growth mode (recursive pattern) be compared to another? Sometimes the comparison can be made quite directly. Consider again the following example from the introduction. We are given the problem of combining the syntactic objects AP, BP, and X^0 via binary Merge. AP and BP are internally complex, while X^0 is a terminal. The options are these:

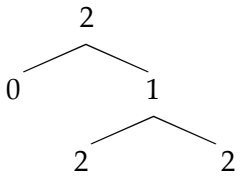
(41) [AP [X^0 BP]] (or [BP [X^0 AP]])

(42) [X^0 [AP BP]]

Again, given just the information that AP and BP are internally complex, the first option produces fewer c-command and containment relations than the second. Noticing the monotonic way in which c-command and containment relations accumulate in a derivation (i.e. additively), this local superiority gives us very good reason for preferring to apply the pattern manifested in the first option over the second more generally, if we are forced to choose one or the other as a repeated format. Put another way, it motivates the choice of the growth mode (43) over (44):

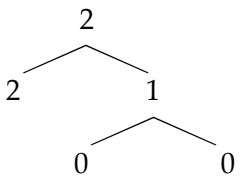
(43) ('X-bar')



(44) 

('high-headed D-bar')

However, this sort of direct comparison will not work for the full comparison set they belong to. Consider another member of that set:

(45) 

('double-headed spine')

No local, direct comparison with the previous two patterns is possible, since they take different inputs (23 calls for two terminals); in general, where (43) and (44) can be applied, (45) cannot.

4.3.3. Indirect Comparison

To get around this problem, I will proceed as follows. First, it is an inescapable fact that these are discrete patterns, ultimately built from some finite number of terminal atoms. This suggests an alternative, slightly indirect way to compare different growth patterns: Compare the set of tree-forms they can generate for some constant number of terminals.

These patterns implicitly define a class of trees. For example, The Spine can be applied to generate (46); that unidirectionally branching structure belongs to the set of trees associated with the growth mode (such a tree can be 'grown' by the pattern). On the other hand, (47) does not belong to the class of trees associated with the Spine.

(46) $[\ W^0\ [\ X^0\ [\ Y^0\ \ Z^0\ ]]]$

(47) $[[\ W^0\ \ X^0\ ]\ [Y^0\ \ Z^0\ ]]$

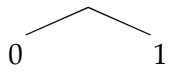
For a fixed number of terminals, there are many different binary-branching arrangements of that number of elements. Some of those branching structures will belong to the class of trees associated with a particular phrase-structure pattern, and some will not. These will typically differ in their number of c-command relations. However, for a fixed number of terminal elements and a particular recursive pattern, we can identify the *best* tree(s), which contain the fewest number of c-command relations of any of the trees associated with a particular pattern. These best trees for a number of terminals then serve as a basis for comparison among the patterns themselves (since, as it turns out, this comparison is monotonic: If a pattern allows a better tree for n terminals than any competing pattern, it also has a better tree for $n+m$ terminals).

4.3.3.1. The 'Bottom of the Tree' Problem

However, this requires some further clarification. The idea is to find some way to compare templates for infinite growth, by isolating them and seeing what happens when they are followed as faithfully as possible. The problem is that none of these rules can be followed completely faithfully. This is an inevitable consequence of insisting that they allow for indefinite recursion: Any such growth pattern must contain 'slots' for other objects of indefinitely large size. Yet the objects which manifest these patterns must ultimately be finite, with nothing but terminal nodes at the bottom of the tree. As a result, some 'slot' that calls for a larger object must be filled with a terminal instead.

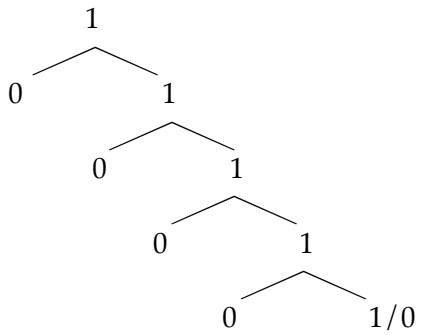
To illustrate, consider the simplest possible growth rule for combining terminals into an indefinitely large recursive structure:

(48)



Even in this, the simplest pattern, the very first step in a derivation presents a problem, as it does not follow the rule. Any derivation whatsoever must begin by creating a structure of the form [X^0 Y^0]; there simply is no other option. So for a pattern like (48), we will accept a structure like (49) as manifesting it as faithfully as possible:

(49)

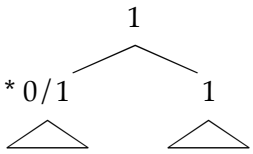


The notation 1/0 indicates where we have deviated from following the growth rule (necessarily, since the tree is finite), here including a terminal where the rule calls for a complex object.

However, if we must allow some 'fudging' at the bottom of the tree, we can at least be faithful everywhere else. Keeping in mind that our ultimate goal is to find some basis for comparing one growth mode to another, we reason that we do not want to 'truncate' the pattern encoded in the growth rule anywhere not required by the brute fact of discreteness. In particular, we will insist that the growth pattern be followed faithfully 'in the middle' of the derivation, so to speak. This amounts to the formal specification that the only deviation from the recursive pattern allowed will be replacing a called-for non-terminal with a terminal. We rule out non-terminal to non-terminal sequencing that violates the pattern, as in (50) below. Here, the notation *0/1 marks the illegitimate portion:

A called-for terminal has been filled with a non-terminal instead.

(50)



4.3.3.2. Top-Down Generation

Note that we have imported a further complication by the convention of assuming that one of the non-terminal types (*n*, the highest of the numbers designating the non-terminal types) will be uniformly associated with the root. Formally, this amounts to generating the trees to be compared from the root down, allowing any branch to terminate. It is an important (if subtle) point that this is not a matter of committing to a top-down view of syntactic derivation, though it should be recognized that a Merge-based system need not be quite so literally bottom-up as often assumed:

> Thus if X and Y are merged, each has to be available, possibly constructed by (sometimes) iterated Merge. […] But a generative system involves no temporal dimension. In this respect, generation of expressions is similar to other recursive processes such as construction of formal proofs. Intuitively, the proof 'begins' with axioms and each line is added to earlier lines by rules of inference or additional axioms. But this implies no temporal ordering. It is simply a description of the structural properties of the geometrical object 'proof'. The actual construction of a proof may well begin with its last line, involve independently generated lemmas, etc. The choice of axioms might come last. (Chomsky 2007a: 6)

Regardless, in the present investigation top-down generation is an artifact of notational choices, rather than a substantive claim.[10] Recall that the objects of interest are recursive cycles. Understood as time-neutral geometric patterns of recursion, these patterns do not properly have a 'beginning' or an 'end' (other than terminal elements, which can in principle appear anywhere in the looping structure as inputs to Merge, but not outputs). Their structure is a matter of how outputs from one step loop into the input to other steps. But we have kept to the familiar tree-diagram notation, assigning numerical designations to non-terminal types. The result is that certain patterns are multiply represented. For example, 'X-bar' and 'high-headed X-bar' are really the same recursive pattern, with a different choice for which non-terminal occurs at the root.

However, it turns out that a certain orientation of the pattern (fixing one or

---

10 In light of this point, the claim made in this article about 'projective structures' needs to be clarified somewhat. Represented in the format [α [β … [γ [X^0 δ]] …]], the claim is a little too strong. What is motivated here is rather the recursive cycle underlying this format. Put another way, even universal strict adherence to such a growth mode in reality would not necessitate that the root node be maximal; the recursive cycle could be oriented differently at the root, thus showing up as one of the 'high-headed' alternatives (such a situation would look like a 'small' projection at the root embedding an otherwise well-behaved projective structure).

another of the non-terminal types at the root) will consistently provide better results than others. Thus for each looping object we can generate a set of alternate versions fixing one or another of the stages as the 'top', corresponding to the 'root' of a tree, and see which are best. Since the basis for comparison is best performance, this should not present a problem in any way.

4.3.3.3. Some Results from Indirect Comparison

Figure 2 graphs the growth in c-command and containment relations for several recursive patterns. Recall that for each growth mode, there is an associated set of trees generated by adhering to the structural pattern consistently from the root down, allowing terminals to appear in 'slots' calling for non-terminals (required for finite trees). For a given number of terminals, a number of trees can be generated by a given pattern. These will differ in the number of c-command and containment relations they encode, but for each choice of growth mode and number of terminals, there will be a best tree (or set of such trees). A 'best tree' has the fewest possible c-command and containment relations that could be produced by that growth mode for that number of terminals. It is these totals which appear in Figure 2 (as a function of the number of terminals).

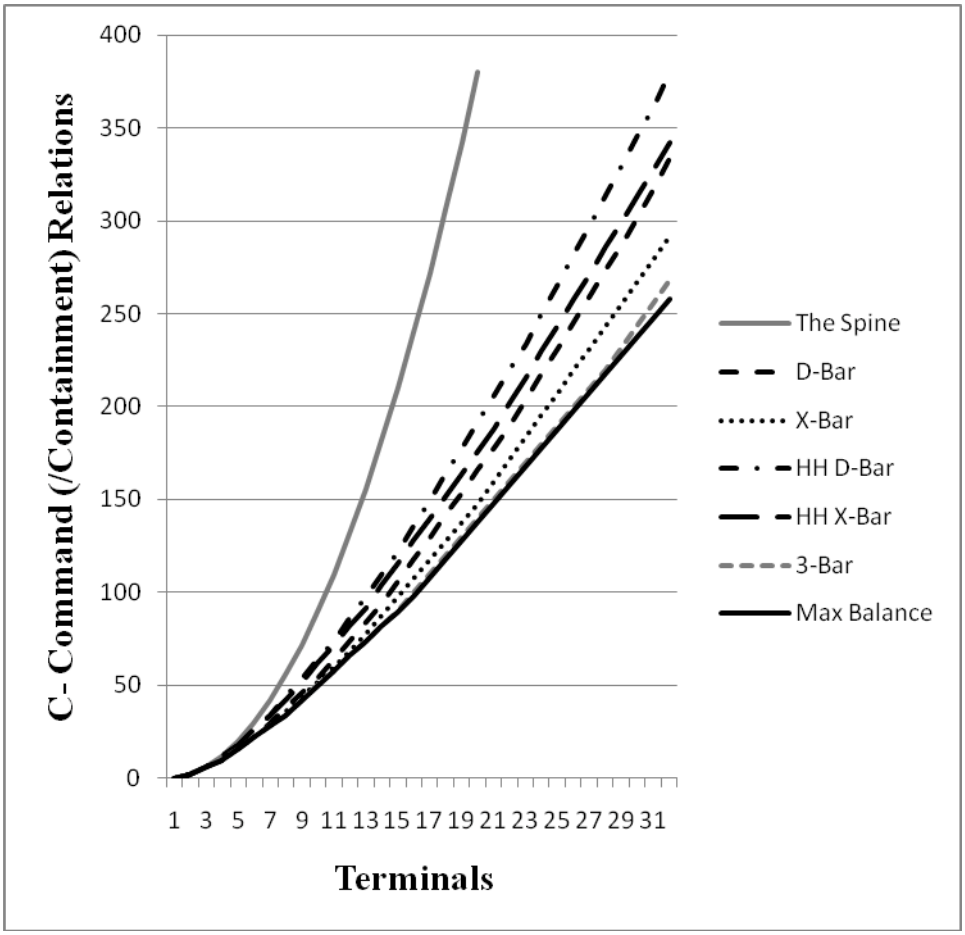


Figure 2: C-command and containment totals as a function of terminals in best trees

I include in the figure 'best trees' in X-bar (34), as well as three other two-layered constructional schemes (35-37). I also include the best system utilizing a 4-way combinatorial distinction (40), which I call '3-bar' (intuitively, an X-bar-like system with two types of intermediate category). The spine (32) forms the upper boundary curve; no growth pattern results in worse performance (in the sense of creating more c-command and containment relations for a given number of terminals). There is also a lower boundary curve, here labeled 'Max Balance'. This is the number of c-command and containment relations in a maximally balanced tree like (26) from section 4.1; the pattern is not the result of any finite growth pattern, but forms the boundary on best-case performance.

Among the growth modes in its comparison set, X-bar has the best performance: Its curve is closer to the best case lower boundary ('Max Balance'). The optimal pattern from the next comparison class, '3-bar', has slightly better performance (the best trees that can be 'grown' by that pattern have fewer c-command and containment relations for the same number of terminals).

To be clear, the figure is meant as an illustration, not a proof. The general result that projective growth modes are best is established formally in the Appendix.

4.4. Deriving Projection

As suggested by Figure 2, X-bar is the best growth mode that can be achieved by any two-stage scheme for constructing recursive structure from terminals via binary Merge. What I call '3-bar' is better still, though it requires more distinctions (more recursive complexity, more information flow) to construct. Generalizing, these are examples of the 'projective' format in (51), where X^0 is a terminal at the 'bottom' of the repeating structure, and α, β, and so on are objects themselves constructed according to (51).

(51) [α [β … [γ [X^0 δ]] …]]

The structural properties of (51) can be captured in our alternate notation as in (52), where 0 is a terminal, and *n* the non-terminal associated with the root.

(52) n → i n
i → j n
…
k → 0 n

The specifier-head-complement format of X-bar theory is one example of such a 'projective structure': Specifically, it is (52) with *n*=2. The more optimal '3-bar' system of (40) is another example, this time with *n*=3. As I prove in the Appendix, this is the optimal format for *n*+1 (i.e. 0, 1, … *n*) types of category (many other less optimal possibilities exist). Intuitively, the idea is as follows. The phrase structural possibilities are understood to be (partially) realized by finite expressions, built bottom-up by Merge. As such, every recursive pattern must include terminals (0s) as one of its structural types. Moreover, no categories are

built solely from non-terminals 'all the way down'.

Given these restrictions, and the determinacy of the structural characterizations assumed, any non-terminal node must dominate a terminal node within depth n, for n+1 types. The best kind of structure, following the format in (52), introduces terminals no closer to the root than forced by this. In essence, introducing terminals too close to the root 'closes off' branches, forcing complex structure to appear deeper in the tree, where it will induce more c-command and containment relations than if it were shallower. The format in (52) allows arbitrarily large structures to be as balanced as possible given the limitations resulting from finitely many structural distinctions.

Note two very interesting properties of (52):

(53) a. Every non-terminal immediately dominates a root-type node.
b. Terminal nodes and root-type nodes are associated one-to-one; a single terminal occurs at the lowest level of the chain of non-root-type nodes dominated by a root-type node.

Replace 'root-type node' with 'maximal projection', 'terminal' with 'head', and 'chain of non-root types dominated by a root type' with 'projection chain', and we have:

(54) a. Every non-terminal immediately dominates a maximal projection.
b. Heads and maximal projections are associated one-to-one; a single head occurs at the lowest level of the projection chain.

That is, the recursive scheme that best minimizes c-command and containment relations has geometric properties corresponding to (54a) the maximality of non-head daughters, and (54b) endocentricity. Such properties are the essence of the theory of projection. But the notions entering into (53) are purely structural ones. Does this 'derive' projection? Not in the sense of literally providing labels on non-terminal nodes. But it suggests a reason for syntactic objects to tend to take the form of structures which are 'ready-made' to be 'read' as projections, in that there is a natural one-to-one association in the optimal format between larger molecules of structure and unique terminals at their 'bottom'.

5. Lexical Features and Projection

This article has been concerned with matters of pure hierarchical geometry, paying no attention to lexical details at all. To say the least, this is orthogonal to the sort of approach pursued in recent work. Following the seminal work of Speas (1990), phrase structure is generally understood to be determined by the specific featural requirements of lexical items, hence 'projected from the lexicon'. Taking this view in conjunction with the principle of Last Resort, which holds that syntactic operations are driven by strict necessity (Chomsky 1995b), there is little room for other principles to play a role in phrase structure. In the strongest version of such a conception, each step in the structure-building process is

required; if some other step were taken, some lexical feature would not be checked appropriately, and the derivation would crash.

5.1. Beyond Features and Last Resort

To be clear, I see no reason to deny that Last Resort accurately describes the mechanisms in play, at an appropriately detailed level of description. But focusing too narrowly on the mechanisms involved may limit the depth of explanation that might be achieved. Consider, for example, the mysterious EPP (Extended Projection Principle) property (Chomsky 1981), which requires that T(ense) must have a filled specifier. This can be enforced by supposing that the relevant head has an EPP feature (or some equivalent device), and that the derivation will crash if the specifier position is not filled. But does this actually explain the EPP, or merely describe it? On this view, it is an accident that T has such a feature; it could just as well have lacked that feature, and then there would be no EPP. What is left unanswered is *why* T should have such a requirement in the first place; can that be explained in some naturalistic way?[11]

The usual approach is to take lexical properties as given *a priori*, with the task of syntax being to accommodate them as best it can. The present approach could be understood as exploring causation in the other direction (i.e. the extent to which syntactic effects might explain lexical properties). Without presupposing that lexical requirements *have* to be what they are, all options are on the table, so to speak. The ultimate goal is to eventually use the insights of the present investigation to achieve a deeper understanding of lexical facts (for example, why an EPP feature for T might be preferred), though this further step is left for future work. That is, independent of the mechanisms which effect structuralization, we may ask about the optimality of the patterns they induce. Insofar as those patterns turn out to be optimal in the sense explored here, they are as expected — 'perfect', Galilean, and explainable in the minimalist mode.

Thus, the point of view here is compatible with even the strictest understanding of how a derivation might be driven by lexical features, if it is allowed that principles of optimality might play some role in determining lexical features.[12] If so, the concerns explored in this article are rather far in the background, indirectly realized through patterns 'frozen into' the lexicon. On the

---

11 Earlier drafts of this work included material showing how concerns of minimizing c-command and containment relations plausibly play a role in displacement, including EPP-movement. The crucial point is that with respect to some of the computations involving such relations at the interfaces, displaced elements are effectively in their displaced position and *not* in their 'base' position (linearization and scope being clear cases), thus opening up the possibility that displacement might serve to derive a more economical form, in the relevant sense. Although the predictions here seem extremely promising, the issues that arise go too far beyond the scope of this article.

12 This may seem odd at first, if 'features' are understood as properties related to interpretation at the interface. Two comments are in order. First, the sorts of features that could most readily be explained by computational considerations are so-called uninterpretable features, which have the dual properties of not being interpreted directly, and seemingly playing a crucial role in structuralization. Second, it may after all be sensible to rethink some properties formerly considered to be properties of interpretation in terms of syntax-internal concerns, pursuing the general program of Hinzen (2006) along these lines.

other hand, in his most recent work Chomsky has suggested that Merge may be driven by a non-specific generalized 'Edge Feature' EF (Chomsky 2007a), which is undeletable in syntax (hence allowing unbounded Merge). If that point of view is adopted, the options for structure are not nearly so rigidly and predictably forced by the choice of lexical items, and the considerations here may play a rather more direct role in determining structure.

5.2. Lexical Requirements: Projection vs. Structuralization

An essential step in the argument is the idea that many possibilities are logically possible for phrase structure. In particular, I argue that we should be willing to be surprised that syntax makes use of 'projective' geometries, wherein terminals occur at the bottom of phrases. But at first glance, this would seem to present a problem: How could it be any other way? That is, if local structures are indeed enforced by lexical requirements, how could a terminal affect structure anywhere except in the sort of domain defined by a projective structure?

What is at stake is the power to enforce structural features, and how far that extends. It is uncontroversial that certain lexical items can force certain structural choices in subsequent Merge operations beyond the first they occur in (i.e. higher up the tree). For example, the item T (Tense), even after Merging with its complement, is able to enforce further details of the derivation, in the form of its EPP property requiring a phrase to occur in its specifier. It seems that T has the reach to place non-local requirements on the structure it occurs in.

But what is required for (lexically-driven) non-projective geometries is that structural enforcement can reach down the tree as well as up. That would require lexical requirements to be discharged 'before' the enforcing head has been Merged; is that not a paradox? Crucially, this sort of situation is not just possible, but empirically attested. The relevant case is long-distance selection: Selectors have the power to enforce properties not just in their complements, but in the interior of their complements as well (thus 'down' the tree).

Boeckx (2008) gives the following example from Hebrew. In Hebrew, as in English, the verb meaning *ask* selects for a [+*wh*] CP. What is important, for our purposes, is the presence of the topicalized phrase *ha sefer* to the left of the [+*wh*] element *le mi*:

(55) Sa'alta oti et ha sefer le mi le haxzir.
asked.2.SG me ACC the book to whom to return
'You asked me to whom to return the book.' (Boeckx 2008: 16)

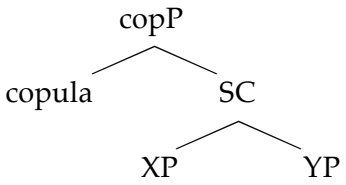
This is a clear case of long-distance selection going beyond strict head–complement. That is, assuming the articulated left periphery of Rizzi (1997), the selected [+*wh*] material occurs embedded inside the phrase hosting the topic. Boeckx notes that Grimshaw's (1991) notion of Extended Projection, or other feature-passing devices, doesn't solve the problem:

> More specifically, it is unclear what it would mean to allow for [+*wh*] information to be 'passed onto' Topic^0, given that [*wh*] marks new

> information, and [Topic] old information. Such a semantic clash of feature composition would be expected to bring the percolation of the relevant feature to an end. (Boeckx 2008: 16)

For another relevant case, consider the analysis of the copula as a head taking a small clause complement (Stowell 1978, Moro 2000), illustrated in (56) below. Here, the lexical copula selects a small clause, an atypical structure resulting from Merge of two full phrases. Small clause structure is a matter of geometry internal to the complement of the copula, hence plausibly another case of long-distance selection in the relevant sense.

(56)



The conclusion seems to be that lexical requirements can enforce structural details down the tree as well as up. This is one reason for carefully separating a notion of 'phrase' tied to projection from a notion involving structure. Surely we do not want to say that portions of the lower clause in the Hebrew example above are 'projections' of the selecting verb, nor that the small clause is a 'projection' of the copula. The point is simply that such an element must occur in a characteristic structure with 'deeper roots', so to speak: Its lexical requirements have structuralization effects that reach down the tree. It should be clear that if such items were the rule, syntactic structures would be drawn from a different set of tree-forms than those described by a projective structure like the X-bar pattern. This only deepens the mystery of why real linguistic structures should tend to adhere to something like the X-bar scheme.

5.3. Antisymmetry and Teleological Reasoning

One important theory which has received little mention throughout is the anti-symmetry approach to linear order following from Kayne (1994). Kayne proposes that linear order is not a primitive relation of the syntactic component, but rather a consequence of certain structural properties. Specifically, he proposes that linear order follows from asymmetric c-command, and derives from this the result that only phrase structures obeying a particularly rigid X-bar format are linearizable.

In a sense, then, Kayne's work may be seen as deriving X-bar shape in terms of PF interface requirements. If one adopts such a view, the concerns explored in this article may seem superfluous, in that X-bar-like structure is 'over-determined' both by interface requirements (antisymmetry) and by minimization of c-command and containment computations (in the present account). But Chomsky has repeatedly emphasized that such a redundancy of explanation should be taken to indicate that some further theoretical reduction is required. Put another way, why would we want to explain the relevant facts in

another way, if antisymmetry already does the job?

Let us examine the antisymmetry account in more detail. One crucial point that must be faced by any version of antisymmetry is that pure geometry as such is insufficient to linearize the result of Merging two complex objects. In Kayne's original work, this problem was resolved using the 'segment/category' distinction proposed by May (1985). That is, when objects XP and YP merge, the result will be, say, YP; in this case, the lower YP is then a 'segment' of the full YP, and does not 'count' for the linearization rule, which is restated in terms of full categories. Notice that this requires both a device of projection (to distinguish whether Merge of XP and YP is an XP or a YP), and some explicit notion of bar-level. Without the latter to distinguish Y^0 and YP, the head Y^0 of YP would be a further segment (with Y') of the larger category (YP), with the undesired result that the X-bar configuration would be an unlinearizable multiple-adjunction structure.

Chomsky's (1995a) Bare Phrase Structure Theory incorporates a similar linearization scheme that improves on this somewhat, in effect recreating the segment/category distinction by insisting that intermediate categories, as neither maximal nor minimal projections, are 'invisible to the computation', a claim which seems empirically supported at least. Nevertheless, projection is still integral to the system, and must be explicitly represented by some device that is visible to the PF component.

In either Kayne's or Chomsky's theory, projection is required for PF demands at least. Without such a device, language would fail to be 'usable' by PF; thus one might argue that antisymmetry could explain projection (and the projective X-bar structure) in terms of requirements imposed by the interface. Why then should any alternative explanation for such properties be countenanced?

I would like to argue that, if we wish to 'explain' syntax naturalistically, we should be suspicious of teleological reasoning of this sort (see especially Hinzen 2006 on this point). That is, supposing that interface conditions — what syntax is 'good for' — explain the mechanisms that syntax has at its disposal is problematic; in a sense, it amounts to a denial of the autonomy of the syntax. Rather, the preferred mode of minimalist explanation is (or ought to be) to explain syntactic facts in terms of concerns internal to the syntax itself. This is the sort of intuition expressed by Uriagereka's notion that it is "as if syntax carved the path that interpretation must blindly follow" (Uriagereka 2002: 64). Thus, whatever the functional necessity of projection, it is something we would like to derive rather than stipulate.

It may be 'good' for language to have a mechanism for projection. But so what? Language, without such a mechanism, would be whatever it was (perhaps unusable, or at least unpronounceable). As Darwin was careful to point out, the 'desire' to fulfill a certain function does not induce internal complexity. This is not the sort of explanation we should be satisfied within the biolinguistic enterprise. It may be 'good' for tigers to have stripes, so that they may be more effectively camouflaged in tall grass, but that does not *cause* them to have stripes. Likewise, the usefulness of flight does not explain how certain creatures come to have the necessary anatomy in the first place. In the case of projection, labels may

be good things to have, but where do they come from?

These concerns are all the more pressing given the biolinguistic perspective, and particularly the rejection of adaptationist accounts. The preferred mode of explanation, if it can be achieved, is to show that properties of language are not just examples of good design, but of *minimal* design as well. Insofar as language seems to have properties that are 'custom-made' for its eventual function, we may feel we have explained more, and in a more satisfying way, if we find that those properties 'emerge' from the optimal functioning of more primitive components. Suppose we have two kinds of accounts for a property like projection. One account appeals to the use to which projection is eventually put to explain why it exists in the first place, in terms of the interpretation (e.g., linearization) of syntactic objects. An alternative account purports to explain projection in an internalist, autonomous way which makes no reference to the eventual use to which language may be put, instead reflecting optimality in 'bare' combinatorics. Then the second account is more of what we are looking for, so to speak. We would prefer to find, not that projection is a principled complication whose mechanism we must stipulate as a primitive, but rather that the mechanism itself is an example of 'order for free', expected to emerge from the optimal operation of more basic components of the system.

5.4. 'Emergent' Projection?

I would like to suggest that this should lead us to rethink the nature of projection in the grammar. Indeed, that concept has a problematic status under current understanding. Simply put, the 'technology' of projection seems to require something non-minimalist, such as assuming that Merge is fundamentally asymmetric, or that Merge necessarily includes a labeling function, both *prima facie* departures from the virtual conceptual necessity of truly 'bare' sets of lexical items. Recognizing this problem, Chomsky suggests that projection is not a primitive notion of syntactic theory, but is to be explained in some way:

> It seems now that much of the architecture that has been postulated can be eliminated without loss, often with empirical gain. That includes the last residues of phrase structure grammar, including the notion of projection or later 'labeling', the latter perhaps eliminable in terms of minimal search.
>
> (Chomsky 2007b: 24)

If the ubiquity of c-command relations in linguistic phenomena reflects a search process (as discussed in section 2), then explaining projection in terms of minimal search is, in fact, exactly what I have proposed. More precisely, what is explained here is why structural correlates of projection are expected in the products of a dynamically optimal derivation. But what is left curiously hanging is the idea of labeling itself: No specific mechanism for enforcing the association between a phrase and its head is motivated. This may well be a good result, given the problems surrounding the technical implementation of labels pointed out by Collins (2002) and others.

In the quote above, it seems that Chomsky has in mind 'eliminating' labeling as a matter of notation explicitly reified by some device in phrase

structure (for example, by complicating the set structure produced by Merge to directly encode the label, as in Chomsky 1995a). But the essential notion remains as a derived fact: Some designated element is required to be readily accessible to determine appropriate interpretation. In the present proposal, a more radical reduction is on offer: The pattern which gives a special place to some designated terminal is independently derived, an accident of optimal branching form. I would like to suggest that this might amount to a deeper explanation; the fact 'emerges' for naturalistic reasons internal to the workings of the computation itself. This is one way of cashing out Uriagereka's notion that it is "as if syntax carved the path that interpretation must blindly follow" (Uriagereka 2002: 64).

6. Conclusions

The Minimalist Program is concerned with the degree to which the abstract mental system that generates syntactic expressions is 'perfect'; that is, as simple and optimal as it could be. This is often cast as the search for explanation in terms of 'virtual conceptual necessity'. But another important facet of minimalist theorizing is that (internal) optimality is also important. This is precisely the nature of 'emergence': Sometimes, very simple systems behaving optimally give rise to complicated structure. In other words, while superficial complexity may seem problematic from the point of view of the minimalist expectation of perfect simplicity, sometimes complex structure is the most perfect solution.

I have argued that the property of 'projection' might be explained in this way. That is, rather than supposing that projection is strictly required for the linguistic system to function at all (a teleological concern which says nothing about where the instantiating mechanism might come from), I argue that a structural basis for projection might emerge from the optimization of unlabeled branching forms. If we suppose that Merge may apply freely, the full spectrum of binary-branching forms is available in principle. But I have argued that there is a computational burden associated with establishing relations based on c-command and containment, such that some derivational choices are better than others. Taking this claim together with the idea that the information which can influence derivational choices is rather local in character, it stands to reason that a syntactic derivation will face the same 'problem' repeatedly, and thus that it might consistently apply the same solution, in the form of a self-similar pattern of recursion. It turns out that the best such patterns correspond to exactly the sort of structures described as 'projections'.

Moreover, there may be reasons for singling out the X-bar pattern of specifier-head-complement from among these projective patterns. The X-bar form has played an important role in linguistic theory for several decades. Here, this pattern has been shown to have properties related to the Fibonacci sequence, a mathematical pattern which pervades nature. It is not much of an exaggeration to claim that patterns related to the Fibonacci sequence are nature's 'favorite solution' to problems of self-similar growth. Of great relevance to the biolinguistic enterprise is the robust, unselected nature of the pattern: Although it is an optimal solution to certain problems, it is apparently not produced by

successive approximations under evolutionary 'tinkering', but emerges robustly and spontaneously from quite general laws of form that shape the inorganic world as well (see Douady & Couder 1992, Thompson 1917/1992, Ball 1999).

I have tried to demonstrate the potential value of considerations orthogonal to another trend in minimalism, the general program of reducing syntactic properties to lexical requirements. One way of understanding the ideas here is as potentially underlying some otherwise mysterious lexical properties, while still maintaining that featural requirements are the mechanism which drives derivational choices. In its strongest form, this proposal could also be taken to indicate a more direct role for hierarchical optimization in determining syntactic forms. In that case, linguistic computation takes on the appearance of a dynamically self-organizing system, and the explanatory burden placed on features and interface requirements is reduced.

This article is one attempt at explaining substantive properties of language in terms of efficient computation and 'laws of form'. This has only been achieved by way of considerable idealization and abstraction; surely the present approach has a long way to go before approaching anything like the rigorous empirical standard to which linguistic research is usually held. It is not clear that detailed predictions are even possible at the level of abstraction here, and it may turn out that nothing more than intriguing analogies will follow from taking it seriously. Even if the specific ideas here prove to be misguided, I hope that the article may at least suggest some new avenues toward deeper explanation of the sort invited by the biolinguistic perspective.

Appendix: Proof of the Optimality of Projective Structure

Take a recursive pattern P to be defined as above over terminal type 0, non-terminal types 1, … *n*, with properties of *determinacy* (every non-terminal i branches according to a unique rule i → j k, with j, k in 0, 1, … *n*), and *termination* (no non-terminal dominates only non-terminals 'all the way down').

The reasoning here will involve the infinite tree-space T generated by maximal iteration of a recursive pattern P. In such trees, every non-terminal node in the recursive pattern will be recursively expanded, and the non-terminals thus introduced will be expanded, and so on 'all the way down'.

Now, we may consider mapping nodes in the tree-space T_1 generated by one pattern P_1 to nodes in the tree-space T_2 generated by another pattern P_2. The idea is to find immediate-containment-preserving maps of sets of nodes in T_1 to sets of nodes in T_2 such that:

(A1) the image of the root node of T_1 is the root node of T_2, and
(A2) if node α immediately contains node β in T_1, the image of α immediately contains the image of β in T_2.

Let us say that T_2 *contains* T_1 if there is some mapping of the set of all nodes

in T_1 into nodes of T_2 meeting this condition, and that T_2 *properly contains* T_1 if T_2 *contains* T_1 but T_1 does not *contain* T_2. (If T_1 *contains* T_2 and T_2 *contains* T_1, then T_1 and T_2 are isomorphic, and so are P_1 and P_2.)

We will also consider finite trees within these infinite trees, i.e. *contained* by them in the sense above. For notational clarity, we reserve T_i for infinite tree-spaces generated by maximal expansion of P_i. Clearly, if T_1 *properly contains* T_2, every finite tree generable by P_2 can be generated by P_1.

We are interested in comparing the optimality, with respect to number of c-command and containment relations, of best finite trees (with equal numbers of nodes) generated by distinct recursive patterns P_1, P_2. At the very least, if every arrangement possible under P_2 is also possible under P_1, but there are arrangements generated by P_1 more optimal than any arrangement of the same number of nodes under P_2, we will judge P_1 to be more optimal than P_2.

(A3) *Lemma 1*
If T_1 *properly contains* T_2, P_1 is more optimal than P_2.

Clearly, every finite tree generable by P_2 can be generated by P_1. For proper containment to hold, T_1 cannot be mapped to T_2. The mapping from T_1 to T_2 fails first at some finite depth d (succeeding at all depths less than d); the maximal finite trees in T_1 and T_2 can be mapped to the other up to depth d–1.

For the mapping to fail, T_1 must have one or more non-terminals at depth d–1 that map to one or more terminals at the same depth in T_2. Then consider the maximal finite tree in T_1 of depth d (all recursive options expanded to depth d, all non-terminals in T_1 at depth d replaced with terminals). This tree has fewer c-command and containment relations than any tree in T_2 with the same number of terminals. One or more of the non-terminals at depth d–1 that were expanded in T_1 must terminate at that depth in T_2. Then some number of nodes in T_1 at depth d cannot be mapped to corresponding nodes in T_2 at the same level, and the same number of nodes must appear at depth d+1 or greater in T_2; all other nodes correspond. Since the number of c-command and containment relations induced by a node is equal to its depth in the tree, it follows that any tree in T_2 containing the same number of nodes as the maximal finite tree of depth d in T_1 must have strictly more c-command and containment relations.

Thus, if T_1 properly contains T_2, P_1 is more optimal than P_2: Every arrangement possible under P_2 is also possible under P_1, but there are arrangements generated by P_1 superior to any arrangement of the same number of nodes under P_2.

(A4) *Lemma 2*
The infinite tree space T_p generated by the projective recursive pattern P_p defined over some number *n* of non-terminal types *properly contains* all tree-spaces T_i generated by distinct recursive patterns P_i defined over the same number of non-terminal types.

To see this, we will need one more concept, that of 'least path-to-terminal'. A 'path' leading from node α to node β is the set of nodes containing α, β, and all

nodes dominating β which are also dominated by α. For any non-terminal node in a tree, we can identify the paths of nodes leading to terminals it dominates, and measure the depth of those paths. Among these paths, there will be one or more least paths-to-terminals (clearly, of depth at most n, for n non-terminal types). Let us consider these paths under the sort of mapping described above.

First, in T_p, the least path-to-terminal from the root node has length *n*. Let us call an 'off-branch' from this path a sub-tree whose root node is immediately dominated by a node on the path, but is not on the path itself. In T_p, the least path-to-terminal from the root of any off-branch is itself of length *n* (since any off-branch is isomorphic to the root node).

Now suppose T_i is a tree-space distinct from T_p defined over the same number n of non-terminal types. First, T_p contains T_i. For this to be false, there must be some finite depth d at which the mapping first fails. Find the shortest path-to-terminal from the root in T_i (or select one of them, if there are several of the same shortest length). Let us map the nodes in this path to nodes in the least path-to-terminal in T_p. This mapping succeeds, because this path is of depth at most n, and the path-to-terminal in T_p is of depth *n*. Now, for each off-branch from the path in T_i, we can map a least path-to-terminal successfully to the least path-to-terminal on the corresponding off-branch in T_p, which again is of the greatest possible depth *n*. And so on, for off-branches of off-branches; this exhausts the set of nodes in T_i, since (due to the termination requirement) every non-terminal lies on some least path-to-terminal. Thus, T_p contains T_i.

It cannot be the case that T_i contains T_p, because we have supposed that T_p and T_i are distinct. Thus, T_p properly contains T_i.

Then from Lemma 1 and Lemma 2, P_p is more optimal than P_i; since P_i was an arbitrary recursive pattern distinct from P_p defined over the same number of non-terminal types, we conclude that the projective pattern is the most optimal.

References

Ball, Phillip. 1999. *The Self-Made Tapestry: Pattern Formation in Nature*. Oxford: Oxford University Press.

Boeckx, Cedric. 2008. *Bare Syntax*. Oxford: Oxford University Press.

Boeckx, Cedric & Massimo Piattelli–Palmarini. 2005. Language as a natural object; linguistics as a natural science. *The Linguistic Review* 22, 447-466.

Bouchard, Denis. 1995. *The Semantics of Syntax: A Minimalist Approach to Grammar*. Chicago: University of Chicago Press.

Bresnan, Joan (ed.). 1982. *The Mental Representation of Grammatical Relations* (Cognitive Theory and Mental Representation). Cambridge, MA: MIT Press.

Carnie, Andrew & David Medeiros. 2005. Tree maximization and the Extended Projection Principle. *Coyote Working Papers in Linguistics* 14, 51-55.

Chametzky, Robert. 1996. *A Theory of Phrase Markers and the Extended Base*. Albany, NY: SUNY Press.

Chametzky, Robert. 2000. *Phrase Structure: From GB to Minimalism* (Generative Syntax 4). Malden, MA: Blackwell.

Chomsky, Noam. 1965. *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press.

Chomsky, Noam. 1970. Remarks on nominalization. In Roderick Jacobs & Peter Rosenbaum (eds.), *Readings in English Transformational Grammar*, 184-221. Waltham, MA: Ginn and Co.

Chomsky, Noam. 1981. *Lectures on Government and Binding: The Pisa Lectures*. Dordrecht: Foris.

Chomsky, Noam. 1995a. Bare phrase structure. In Gert Webelhuth (ed.), *Government and Binding Theory and the Minimalist Program* (Generative Syntax 1), 385-439. Malden, MA: Blackwell.

Chomsky, Noam. 1995b. *The Minimalist Program* (Current Studies in Linguistics 28). Cambridge, MA: MIT Press.

Chomsky, Noam. 2000. Minimalist inquiries: The framework. In Roger Martin, David Michaels & Juan Uriagereka (eds.), *Step by Step: Essays on Minimalist Syntax in Honor of Howard Lasnik*, 89-155. Cambridge, MA: MIT Press.

Chomsky, Noam. 2001. Derivation by phase. In Michael Kenstowicz (ed.), *Ken Hale: A Life in Language* (Current Studies in Linguistics 36), 1-52. Cambridge, MA: MIT Press.

Chomsky, Noam. 2005. Three factors in language design. *Linguistic Inquiry* 36, 1-22.

Chomsky, Noam. 2007a. Approaching UG from below. In Uli Sauerland & Hans-Martin Gärtner (eds.), *Interfaces + Recursion = Language? Chomsky's Minimalism and the View from Syntax–Semantics* (Studies in Generative Grammar 89), 1-29. Berlin: Mouton de Gruyter.

Chomsky, Noam. 2007b. Of minds and language. *Biolinguistics* 1, 9-27.

Chomsky, Noam & Morris Halle. 1968. *The Sound Pattern of English*. New York: Harper and Row.

Cinque, Guglielmo. 1999. *Adverbs and Functional Heads: A Cross-linguistic Perspective* (Oxford Studies in Comparative Syntax). New York: Oxford University Press.

Collins, Chris. 2002. Eliminating labels. In Samuel David Epstein & T. Daniel Seely (eds.), *Derivation and Explanation in the Minimalist Program* (Generative Syntax 6), 42-64. Malden, MA: Blackwell.

Douady, Stephane & Yves Couder. 1992. Phyllotaxis as a physical self-organized process. *Physical Review Letters* 68, 2098-2101.

Epstein, Samuel David, Erich Groat, Ruriko Kawashima & Hisatsugu Kitahara. 1998. *A Derivational Approach to Syntactic Relations*. Oxford: Oxford University Press.

Freidin, Robert & Jean-Roger Vergnaud. 2001. Exquisite connections: Some remarks on the evolution of linguistic theory. *Lingua* 111, 637-666.

Gazdar, Gerald, Ewan Klein, Geoffrey Pullum & Ivan Sag. 1985. *Generalized Phrase Structure Grammar*. Cambridge, MA: Harvard University Press.

Grimshaw, Jane. 1991. Extended projection. Ms., Brandeis University.

Halle, Morris & Jean-Roger Vergnaud. 1987. *An Essay on Stress* (Current Studies in Linguistics 15). Cambridge, MA: MIT Press.

Hayes, Bruce. 1995. *Metrical Stress Theory*. Chicago: University of Chicago Press.
Hinzen, Wolfram. 2006. *Mind Design and Minimal Syntax*. Oxford: Oxford University Press.
Hornstein, Norbert & Jairo Nunes. 2008. Adjunction, Labeling, and Bare Phrase Structure. *Biolinguistics* 2, 57-86.
Jackendoff, Ray. 1977. *X'-Syntax: A Study of Phrase Structure* (Linguistic Inquiry Monographs 2). Cambridge, MA: MIT Press.
Kayne, Richard S. 1984. Unambiguous paths. In Richard S. Kayne, *Connectedness and Binary Branching*, 129-163. Dordrecht: Foris.
Kayne, Richard S. 1994. *The Antisymmetry of Syntax* (Linguistic Inquiry Monographs 25). Cambridge, MA: MIT Press.
Kornai, Andras & Geoffrey Pullum. 1990. The X-bar theory of phrase structure. *Language* 66, 24-50.
May, Robert. 1985. *Logical Form* (Linguistic Inquiry Monographs 12). Cambridge, MA: MIT Press.
Moro, Andrea. 2000. *Dynamic Antisymmetry* (Linguistic Inquiry Monographs 38). Cambridge, MA: MIT Press.
Pietroski, Paul. 2004. *Events and Semantic Architecture*. Oxford: Oxford University Press.
Pollard, Carl & Ivan Sag. 1987. *Information-Based Syntax and Semantics*, vol. 1 (CSLI Lecture Notes 13). Stanford: CSLI Publications.
Pollard, Carl & Ivan Sag. 1994. *Head-Driven Phrase Structure Grammar*. Chicago: University of Chicago Press.
Reinhart, Tanya. 1976. *The Syntactic Domain for Anaphora*. Cambridge, MA: MIT dissertation.
Rizzi, Luigi. 1990. *Relativized Minimality* (Linguistic Inquiry Monographs 16). Cambridge, MA: MIT Press.
Rizzi, Luigi. 1997. The fine structure of the left periphery. In Lilliane Haegeman (ed.), *Elements of Grammar: Handbook of Generative Syntax* (Kluwer International Handbooks of Linguistics 1), 281-337. Dordrecht: Kluwer.
Speas, Margaret. 1990. *Phrase Structure in Natural Language* (Studies in Natural Language & Linguistic Theory 21). Dordrecht: Kluwer.
Stowell, Tim. 1978. What was there before there was there? In Donka Farkas, Wesley Jacobson & Karol W. Todrys (eds.), *Papers from the Fourteenth Regional Meeting of the Chicago Linguistic Society*, 458-471.
Stowell, Tim. 1981. *Origins of Phrase Structure*. Cambridge, MA: MIT dissertation.
Thompson, D'Arcy Wentworth. 1917/1992. *On Growth and Form* [abridged edn., prepared by John Tyler Bonner]. Cambridge: Cambridge University Press.
Uriagereka, Juan. 1998. *Rhyme and Reason: An Introduction to Minimalist Syntax*. Cambridge, MA: MIT Press.
Uriagereka, Juan. 2002. *Derivations: Exploring the Dynamics of Syntax* (Routledge Leading Linguists). London: Routledge.
Williams, Edwin. 1994. *Thematic Structure in Syntax* (Linguistic Inquiry Monographs 23). Cambridge, MA: MIT Press.

David P. Medeiros
University of Arizona
Department of Linguistics
P.O. Box 210028
Tucson, AZ 85721
USA
medeiros@email.arizona.edu