Abstract

Aravind Joshi famously hypothesized that natural language syntax was characterized (in part) by mildly context-sensitive generative power. Subsequent work in mathematical linguistics over the past three decades has revealed surprising convergences among a wide variety of grammatical formalisms, all of which can be said to be mildly context-sensitive. But this convergence is not absolute. Not all mildly context-sensitive formalisms can generate exactly the same stringsets (i.e. they are not all weakly equivalent), and even when two formalisms can both generate a certain stringset, there might be differences in the structural descriptions they use to do so. It has generally been difficult to find cases where such differences in structural descriptions can be pinpointed in a way that allows linguistic considerations to be brought to bear on choices between formalisms, but in this paper we present one such case. The empirical pattern of interest involves wh-movement dependencies in languages that do not enforce the wh-island constraint. This pattern draws attention to two related dimensions of variation among formalisms: whether structures grow monotonically from one end to another, and whether structure-building operations are conditioned by only a finite amount of derivational state. From this perspective, we show that one class of formalisms generates the crucial empirical pattern using structures that align with mainstream syntactic analysis, and another class can only generate that same string pattern in a linguistically unnatural way. This is particularly interesting given that (i) the structurally-inadequate formalisms are strictly more powerful than the structurally-adequate ones from the perspective of weak generative capacity, and (ii) the formalism based on derivational operations that appear on the surface to align most closely with the mechanisms adopted in contemporary work in syntactic theory (merge and move) are the formalisms that fail to align with the analyses proposed in that work when the phenomenon is considered in full generality.

1. Introduction

Joshi (1985) famously hypothesizes a computational characterization of natural language: the mildly context-sensitive (MCS) languages are those satisfying the following three properties:

- limited cross-serial dependencies: the class of natural languages should be sufficiently expressive to characterize the cross-serial word order patterns found in languages like Swiss German (Shieber 1985).
constant growth: the sentences in any natural language should not have gaps of unbounded length (i.e., if we order the strings in a language by length, the gaps should be bounded in size).

polynomial parsing complexity: the class of natural languages should permit an efficient algorithm for assigning structural descriptions to strings.

The intuition behind the MCS proposal was much the same as the one underlying generative linguistics: a “reasonable” theory of human language should provide both a lower bound on how expressive a formalism for natural language needs to be (the cross-serial dependency property) while also providing limits on the range of patterns that can be characterized (the constant growth and polynomial parsing complexity properties).

Joshi’s proposal spawned an important line of work that identified a varied set of grammar formalisms all satisfying the MCS properties. These included Tree Adjoining Grammars (TAGs), Linear Indexed Grammars (LIGs), Head Grammars (HGs) and Combinatory Categorial Grammars (CCGs) (Joshi, Vijay-Shanker and Weir, 1991). In fact in addition to sharing all the MCS properties, TAGs, LIGs, HGs and CCGs have identical “weak generative capacity”: they can characterize precisely the same sets of sentences (strings).

Other work has shown that Multi-Component TAG, Multiple Context-Free Grammars (MCFGs; Seki et al., 1991) and Minimalist Grammars (MGs; Stabler 1997) also satisfy the MCS properties. Interestingly, they do this while being strictly more expressive in weak generative capacity, as illustrated in the diagram in (1): they can define all the sets of sentences that TAG/CCG/LIG/HG can, and some that TAG/CCG/LIG/HG cannot.\(^1\)

\(^1\) The reason this is possible is because the upper-bound provided by constant-growth and polynomial parsing does not uniquely determine the TAG-equivalent class. In fact, Joshi (1985) suggested an interpretation of the limited cross-serial dependencies property according to which it imposes limits on the expressivity of natural language. In particular, he suggested that natural language grammars should not be able to express “permutation closure”. This would exclude from consideration languages that instantiate patterns like the one found in the string set MIX = \{w ∈ \{a,b,c\}∗ | #_a(w) = #_b(w) = #_c(w)\}. Subsequent work has found that MIX is generable by MCFGs (and the weakly equivalent MGs) leading Steedman (2018) to argue against their use as a part of a theory of natural language syntax. Instead, he advocates for a more restrictive hypothesis about the computational nature of language that aligns with the original TAG/LIG/HG/CCG class: the Slightly Non-Context-Free (SNCF) languages.
At the same time, it is important to note that even if we limit our attention to stringsets that can be characterized by any of these formalisms, the same stringset might be generated by different formalisms in rather different ways. So linguistic phenomena that fall within the shared region by the metric of weak generative capacity might receive analyses that differ in linguistically important ways, under different formalisms. In this paper we'll present such a case, where TAG and LIG allow for analyses of a certain pattern that appear to be more linguistically appropriate than the analyses that are possible under MG and MCFG, despite the fact that the latter are more powerful from the perspective of weak generative capacity.

Because of incomparability in the data structures posited by various formalisms, it has proven elusive to develop a precise and broadly-applicable notion of “strong generative capacity” — one that would allow us to draw a diagram with a region reflecting the range of structural patterns each formalism can generate, and claim that a certain pattern of interest lies outside the MG/MCFG region but inside the TAG/LIG region of that diagram.³

² But cf. Miller 1999 for a useful attempt.
³ The crucial point about structural descriptions that we will be appealing to is underlyingly based on the fact that a certain tree language is generable by monadic linear context-free tree grammars (MLCFTGs) but is not generable by multiple regular tree grammars (MRTGs). The tree languages generable by MLCFTGs bear an important relationship to the structures derived by TAGs (Kepser and Rogers, 2011), and those generable by MRTGs bear an important relationship to the structures derived by MGs (Kobele et al. 2007, Mönnich 2007); the two classes are incomparable. The relevant differences between MLCFTGs and MRTGs correspond precisely to the differences between TAG/LIG and MG that we focus on in this paper. However, the differing linguistic assumptions that are typically made in TAG, LIG and MG mean that the derived trees generated by each formalism do not use a common vocabulary of symbols, even when the same line of analysis is assumed. (It is not even clear that there is a single identifiable notion of “derived tree” in MGs.) As a result, it is not a straightforward matter to directly compare the derived tree sets across formalisms. Instead we focus in this paper on the intuitive properties that distinguish TAG, LIG and MLCFTG on the one hand from MG and MRTG on the other, without fleshing out the connections in formal detail.
In this paper we will take a more modest, targeted approach to assessing and comparing the structural properties of different kinds of grammars: we identify a coarse-grained classification of formalisms along two dimensions, Ext and Fin, that highlight the distinctive and intuitively relevant points of variation in their structure-building mechanisms. These dimensions, which stem from central ideas in current minimalist syntax, are as follows:

- The extension condition (Ext) proposed by Chomsky (1995) embodies a cyclicity constraint on syntactic derivations: the syntactic structure can grow only at (or near) the root.
- Conditions like Phase Impenetrability (Chomsky 2000) and Shortest Move (Rizzi 1990) can be understood as limiting how much information about the derivational past, what we call derivational state, can condition the applicability of a subsequent operation. Some versions of these conditions result in a fixed finite bound on the amount of derivational state, and we call this a finiteness condition (Fin).

As we will see in detail below, Ext and Fin define points of variation across MCS grammar formalisms. On the one hand, Minimalist Grammars (MGs; Stabler 2011) abide by Ext, making use of the standard minimalist operations of Merge and Move. So too do Combinatory Categorial Grammars (CCGs; Steedman 1996), whose derivations involve bottom-up concatenation of lexical items, and the closely related Linear Indexed Grammars (LIGs; Gazdar 1988). In contrast, the adjoining operation of Tree Adjoining Grammars (TAGs; Joshi and Schabes 1997) allows trees to grow “in the middle,” which means that TAG derivations may violate Ext. Turning to Fin, both MG and TAG operate with bounded derivational state, whereas CCG, with its unboundedly large (derived) categories, and LIG, with its stack-valued non-terminals, permit unbounded state. These relationships are summarized in the table in (2).

<table>
<thead>
<tr>
<th></th>
<th>Ext</th>
<th>¬Ext</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fin</td>
<td>MG, MCFG</td>
<td>TAG</td>
</tr>
<tr>
<td>¬Fin</td>
<td>LIG, CCG</td>
<td></td>
</tr>
</tbody>
</table>

In the first part of this paper (sections 2 through 4), we show how these dimensions impact the analysis of a specific phenomenon involving multiple wh-movement dependencies, one which is known to induce grammatical complexity (Miller, 1991). We will see that the standard minimalist analysis of this phenomenon is incompatible with a certain part of this classificatory space, where both Ext and Fin hold.

The upshot from this conclusion is that we can write down a TAG or LIG/CCG that generates a certain structural pattern, corresponding to a standard syntactic analysis for the construction under consideration, that no MG or MCFG can generate. This allows us to argue against the viability of the standard version of the MG formalism. Yet this result is striking in the face of the
well-studied weak generative capacity relationships between these formalisms: the “structurally inadequate” formalisms are more expressive in terms of characterizing sets of strings than the “structurally adequate” ones. This leads to a natural follow-up question: what sort of structures does a MG/MCFG use to generate the string pattern that arises from what we have claimed to be the “right” structures? In principle we might find that the structural descriptions used by this MG/MCFG, while unfamiliar, are linguistically plausible in their own right, or perhaps even preferable in some way to existing practice. In the second part of the paper (section 5), we demonstrate that the structural descriptions that an MG/MCFG must resort to appear to have little to recommend them.

2. The Bulgarian data, the “everyday minimalist” account, and unbounded derivational state

2.1 The data

Our empirical starting point is Bulgarian questions like (3) and (4) (Rudin 1988; Richards 1997):

(3) 
Koja kniga te popita učitelja [ko[2 ubedi Ivan t[1 da publikiva t[1]]]?

which book you asked teacher who convinced Ivan to publish

“Which book did the teacher ask you who Ivan convinced to publish?”

(4)
Koj kontinent te popita učitelja [koj t[2 e otkril t[1]]?  
which continent you asked teacher who has discovered

“Which continent did the teacher ask you who discovered?”

Both of these examples involve dependencies between two wh-phrases and their corresponding trace positions. Descriptively, the crucial configuration is the one that shows that languages like Bulgarian do not enforce the wh-island constraint (although its relationship to a general theory of islands is not our concern here). We assume that this empirical pattern can be extended without any bound on the number of displaced wh-phrases, as indicated in (5), modulo performance complications (Miller and Chomsky 1963, but cf. Joshi, Becker, and Rambow 2000).

(5) 2 wh-phrases: [wh …[wh […]t …t …]]

3 wh-phrases: [wh …[wh …[wh […]t …t …t …]]]

4 wh-phrases: [wh …[wh …[wh …[wh […]t …t …t …t …]]]]

From the perspective of the word sequences alone, the pattern under discussion, where a certain number of wh-phrases must be followed by an equal number of traces, is not unusually complex: a context-free grammar (CFG) can achieve this result. A CFG would however necessarily do so by assigning the following structural descriptions to the sentences, abstracting away from irrelevant details:
In such structures, the wh-elements and traces form a sequence of center-embedded dependencies: the innermost wh-element and trace form a constituent that is embedded within another that contains the next innermost wh-phrase and trace, etc. However, linguists analyzing sentences like (3) and (4) have uniformly assumed a rather different kind of structure, which is schematically depicted in (7).

The crucial property that these structures have is that all of the traces are contained in a single constituent that includes none of the corresponding wh-phrases. Note that this point about constituency is at least partially independent of how the dependencies between the wh-phrases and traces are organized: the structure in (7) is in principle compatible with both nested or crossing configurations, though there might be empirical reasons to favor one or the other (Pesetsky 1982, Richards 1997). As a result, we do not include any subscripts in (6) and (7). Nonetheless, enforcing the requirement that the number of wh-phrases must be identical with the number of traces, in the face of the structural description in (7), is simply not possible with context-free grammars.

### 2.2 Some familiar derivational strategies

Let us now consider the nature of a minimalist-style derivation for example (4). Under usual assumptions, derivations proceed in a bottom-up (Ext-satisfying) fashion, in a way that reflects both the internal structure of the clause and the selectional properties of the predicates that are
involved. This gives rise to the derivation depicted informally in (8). Each step of this derivation corresponds to one or more instances of Merge (a, b, d) or an instance of Move (c, e). Highlighting indicates phrases with unchecked featural requirements of the sort that create the crucial wh-dependencies.

At the right of each derivational step, we indicate the number of highlighted elements. These unchecked features contribute to the derivational state in that they condition subsequent grammatical operations: each instance of an unchecked feature in the derivational state means that another element with a corresponding feature must be introduced to satisfy the unchecked feature. The important point is that in step (b), all of the derivation’s wh-phrases (here, two) have unchecked featural requirements. Assuming that the relevant pattern extends without bound, this means that there will be no bound on the number of highlighted elements, and therefore the amount of derivational state, that a derivation might need to track. As a result, a minimalist derivation that can produce such structures will violate Fin.

The Phase Impenetrability Condition (PIC) is often thought to impose a limit on the amount of information accessible to the derivation, and therefore might result in Fin being satisfied. However, imposing the PIC on the derivation in (8) and requiring successive cyclic movement, as in (9), does not change the upper limit on how many phrases must be “highlighted” during the derivation.\(^4\)

The PIC constrains \textit{where} the highlighted things are allowed to be, but not \textit{how many} phrases can be highlighted at a single point: as long as there is no limit on the number of specifiers of a particular CP, there will be no limit to how many moving phrases can fit into the region that remains accessible when the PIC is imposed.\(^5\) So, with or without the PIC, in step (b), \textit{all} of the

\(^4\) For ease of exposition we follow a strict version of Ext in (9) and do not assume “Tucking In” (Richards, 1997) in step (d), but the important point is unaffected by this.

\(^5\) A limit on the number of phrases with features to be subsequently checked would be imposed if the PIC was combined with a limit on the number of positions in the syntactic structure between two phase heads. This would follow if there was a bound on the number of syntactic heads that can be merged in a single phase (derivable from argument structure considerations, perhaps) together with the assumption that a given syntactic head can host only a bounded number of elements in its projection. The resulting system would be unable to generate the Bulgarian pattern under discussion in a natural way, for essentially the same reason that the MG formalism we discuss in section 5 below is unable to. But a standard working assumption in the
derivation’s wh-phrases have remaining unchecked featural requirements. Consequently, as the pattern is extended, either of these two Ext-satisfying strategies (i.e., (8) and (9)) will require unbounded derivational state (i.e., ¬Fin).

Before we proceed, let us point out that the pattern derived in (10) is importantly different from the one considered to this point.

(10) Who do you think t₁ wonders what t₂ John bought t₂ yesterday?

In this example, each wh-phrase is introduced and moved to its surface position before the next one is merged. As a result, derivations of this kind do not require unbounded derivational state as the number of wh-phrases is increased.

3. The trade-off between Ext and Fin

The discussion in the previous section showed how an Ext-satisfying derivation for a certain kind of structure will necessarily violate Fin. However, as we will see below, this is not the only conceivable approach. If for example one chooses a derivational strategy that violates Ext, it then becomes possible to generate the structural pattern in a way that respects Fin. Before demonstrating this non-Ext and Fin derivational strategy, as formally instantiated in TAG (section 3.2), and comparing to the formal instantiation of Ext and non-Fin derivations in LIG and CCG (section 3.3), we first turn to a discussion of the abstract trade-off that holds between Ext and Fin (section 3.1).

In section 4 we will see that while LIG and CCG formally instantiate the side of this trade-off that corresponds to the familiar minimalist-style derivations discussed above, this non-Fin and Ext strategy is importantly different from what the MG formalism allows.

3.1 The trade-off, abstractly

Suppose we need to construct an arrangement of white and black pebbles in a line, so that there are an equal number of each, with all the white pebbles preceding all the black ones. One can imagine a variety of ways of doing this, of which two seem particularly canonical. In the first of these, we proceed from left to right, always extending the line of pebbles at its (right) end. We maintain a counter to keep track of the number of white pebbles already laid down. Once we start laying down black pebbles, we reduce the counter by one, and we conclude when the counter returns to zero.

face of data like (3) and (4) is that the CP projections can have unboundedly many specifiers (Richards, 1997), in violation of the second assumption just mentioned. If this is the case, then even though the PIC requires that all moving phrases be in the region consisting of the current phase plus the periphery of the last phase, there is no limit to how many movers can be maintained in that region.
Work from one end to the other (≈ Ext), unbounded memory (¬ Fin)

![Diagram](image)

Working at one end of the pebbles is a sequential analog to our Ext property. Because there is no bound on the number of white pebbles that can be laid down, there is no bound on the value of the counter. This corresponds then to unbounded derivation state, i.e., ¬ Fin.

The alternative derivation eschews the left-to-right construction of the pebble sequence, but instead alternates the placement of a white pebble to the left of the sequence so far created with the placement of a black pebble to the right of the sequence.

Work inside-out/outside-in (≈ ¬ Ext), finite memory (Fin)

![Diagram](image)

The possibility of working at both ends of the sequence constitutes a violation of the string analog of Ext. However, doing so means that we no longer need to maintain a counter, or at least not one which counts any higher than 1. Thus this strategy satisfies Fin.

Interestingly, the relationship between this pair of strategies is similar to the one between CFGs and Push-Down Automata (PDAs). A PDA processes a sequence of a certain number of as (white pebbles) with the same number of bs (black pebbles) from one end of the string to the other using an unbounded stack-based memory to ensure that the number of as and bs is identical. The PDA instructions are given here, together with a corresponding derivation:

(14)

a. \((\cdots, \cdots) \Rightarrow (\cdots a, \cdots X)\) ("read an a, push an X")

\((\cdots, \cdots X) \Rightarrow (\cdots b, \cdots)\) ("read a b, pop an X")

b. \((a, X) \Rightarrow (aa, XX) \Rightarrow (aaa, XXX) \Rightarrow (aaab, XX) \Rightarrow (aaab, X) \Rightarrow (aaabb, )\)

Being restricted to reading through a string from one end to the other, a PDA must use its unbounded stack to maintain a count of unmatched occurrences of a, and this count can grow without bound. Because of the unbounded number of Xs that occur during the derivation, like the unbounded counter in (12), this PDA derivation is ¬ Fin (and indeed restricting PDAs to having only finite memory limits their expressive power to regular languages).

In contrast, a CFG generates the same sequences of the form \(a^n b^n\) using an outside-in derivational regimen and finite derivational state. Beginning with a start symbol S, the first rule of the grammar generates an a at the left of the string’s midpoint, which is now relabeled X (indicating a surplus of one a). The X then triggers the addition of a b to the right of the string’s midpoint, leaving behind an S at the midpoint (indicating an equal number of as and bs). A sample derivation is given immediately below.

(15)

a. \(S \Rightarrow a X\)

\(X \Rightarrow S b\)

\(S \Rightarrow \varepsilon\)

b. \(S \Rightarrow a X \Rightarrow a S b \Rightarrow a a X b \Rightarrow a a S b b \Rightarrow a a a X b b \Rightarrow a a a S b b b \Rightarrow a a a b b\)
Because the rewriting takes place in the middle of the string, this derivation violates (the string-generation analog of) \( \text{Ext} \). However, the fact that the rewriting sequence \( S \Rightarrow aX \Rightarrow aSb \) introduces an \( a \) along with its corresponding \( b \), only a bounded amount of derivational state is needed, satisfying \( \text{Fin} \): as in (13), no record of how many \( a\)-\( b \) pairs have been introduced needs to be maintained to condition subsequent rewrites.

### 3.2 A \( \neg \text{Ext}, \text{Fin} \) Derivational System: Tree adjoining grammar (TAG)

Let us now move to examining the \( \text{Ext}/\text{Fin} \) trade-off in the context of the construction of trees rather than strings. The Tree Adjoining Grammar (TAG) formalism constructs trees through the combination of a finite set of elementary trees that constitute a grammar. TAG provides two modes of combination, substitution and adjoining. Substitution augments an existing piece of structure by inserting it into a position along the frontier of another elementary tree. In contrast, the adjoining operation adds to a tree by inserting a recursive piece of structure, called an auxiliary tree, at an internal node, thereby violating \( \text{Ext} \). (Non-auxiliary trees are called initial trees.) This mechanism can be used for the introduction of modification structures. In (16), a VP-recursive auxiliary tree is adjoined to a mono-clausal structure, which is expanded at the VP node, to derive the structure shown on the right.

\[(16)\]

\[
\begin{array}{c}
\text{TP} \\
\text{John} \\
\text{VP} \\
\text{saw} \quad \text{Mary} \\
\text{VP} \quad \text{PP} \\
\text{on} \quad \text{Tuesday} \\
\end{array}
\quad \quad
\begin{array}{c}
\text{TP} \\
\text{John} \\
\text{VP} \\
\text{VP} \\
\text{PP} \\
\text{saw} \quad \text{Mary} \quad \text{on} \quad \text{Tuesday} \\
\end{array}
\]

Adjoining can also be used to achieve the effects of unbounded wh-movement (Kroch 1987, Frank 2002), by adjoining clausal structure between a wh-element, which was fronted in its elementary tree, and the remainder of its clause:

\[(17)\]

\[
\begin{array}{c}
\text{CP} \\
\text{what}_1 \\
\text{C'} \\
\text{TP} \\
\text{John} \quad \text{VP} \\
\text{bought} \quad t_1 \\
\end{array}
\quad \quad
\begin{array}{c}
\text{CP} \\
\text{what}_1 \\
\text{C'} \\
\text{TP} \\
\text{you} \quad \text{think} \quad C' \\
\end{array}
\]

\[
\begin{array}{c}
\text{CP} \\
\text{what}_1 \\
\text{C'} \\
\text{TP} \\
\text{you} \quad \text{think} \quad C' \\
\text{bought} \quad t_1 \\
\end{array}
\]
The steps in a TAG derivation are conditioned only by the elementary trees that are being combined. Since a grammar consists of only finitely many elementary trees, each of bounded size (cf. the *Condition on Elementary Minimality* from Frank 2002), the identity of an elementary tree constitutes a finite amount of derivational state that must be maintained. These derivations therefore satisfy **Fin**.

Now let us return to the pattern from (5) that has motivated our discussion. In fact, TAG can generate the pattern in (5) using a form of the $\neg \text{Ext}, \text{Fin}$ strategy discussed above for strings. Elementary trees introduce matched pairs of a wh-phrase and a c-commanded trace, as shown in (18) for example (3), just like the $a$-$b$ pairs introduced by the CFG rewrites in (15). The fact that the trees are not constrained to grow only at one end ($\neg \text{Ext}$) allows the tree-building system to operate with a finite amount of memory (**Fin**).

\[(18)\]

\[
\text{CP} \quad \text{TP} \\
\text{C} \quad \text{VP} \\
\text{teacher} \quad \text{CP}
\]

\[
\text{CP} \quad \text{TP} \\
\text{C} \quad \text{VP} \\
\text{ask} \quad \text{CP}
\]

\[
\text{CP} \quad \text{TP} \\
\text{C} \quad \text{VP} \\
\text{Ivan} \quad \text{CP}
\]

\[
\text{CP} \quad \text{TP} \\
\text{C} \quad \text{VP} \\
\text{PRO} \quad \text{VP}
\]

\[
\text{CP} \quad \text{TP} \\
\text{C} \quad \text{VP} \\
\text{publish} \quad \text{t}_1
\]

3.3 **Ext, $\neg \text{Fin}$ Derivational Systems: Linear Indexed Grammars (LIG) and Combinatory Categorial Grammars (CCG)**

Linear Indexed Grammars (LIGs) add to CFGs the ability to store derivational state not only in the identity of the non-terminal symbol that is expanded during the derivation but also in the contents of an unbounded stack at each node of a tree. This information can be propagated through a vertical path in a tree in order to control non-local dependencies, such as the dependency between a wh-phrase and its trace. While LIG derivations construct trees in the same end-to-end (**Ext**) fashion as CFG derivations, LIG has an unbounded (and hence $\neg \text{Fin}$) amount of stack-based memory available to ensure that wh-phrases and c-commanded traces are appropriately paired up — just as a PDA uses its stack to pair occurrences of $a$ and $b$ when generating $a^n b^n$. Thus, we can introduce LIG rules which ensure that an $X$ is added to the stack whenever a wh-dependency is introduced, and that an $X$ is removed when the wh-dependency is resolved:

\[(19)\]

\[
\text{CP}[\ldots] \rightarrow \text{WH C'}[X\ldots]
\]

\[
\text{DP}[X\ldots] \rightarrow t
\]

Assuming that the other rules of the grammar leave the stack unchanged, this will result in the generation of long-distance wh-dependencies as follows:
This same derivational strategy generalizes to cases like (3). The derivation proceeds in a consistent end-to-end (Ext) fashion, but allows the derivational state to grow unboundedly with the number of wh-phrases.
Combinatory Categorial Grammar (CCG) is closely analogous to LIG in the relevant respects: CCG derivations construct trees in an \textbf{Ext}-respecting fashion, and allow the retention of unboundedly large derivational state ($\neg\textbf{Fin}$) through the construction of unboundedly large complex categories.

CCG uses complex categories, formed from a finite number of atomic categories by forward and backward slashes, to encode the combinatory potential of an expression. The basic use of these slashes follows the general rules in (22). A simple illustrative derivation is shown in (23).

\begin{equation}
\frac{X/Y}{X} \quad > \quad \text{forward function application}
\end{equation}

\begin{equation}
\frac{Y \quad X\backslash Y}{X} \quad < \quad \text{backward function application}
\end{equation}
In addition to the basic function application rules in (22), the function composition rules in (24) allow complex categories built out of these same slashes to express non-local dependencies.

(24) Function composition rules

\[
\frac{X/Y \quad Y/Z}{X/Z} \quad \text{forward function composition}
\]

\[
\frac{Y\backslash Z \quad X\backslash Y}{X\backslash Z} \quad \text{backward function composition}
\]

In (25), the forward function composition rule allows John to be combined with bought in a way that leaves the rightward NP argument position unfilled, and a single instance of “wh-movement” is generated through the propagation of this rightward NP argument (encoded as “/NP”) on all of the derived categories on the path between the verb and the wh-phrase. The subexpressions John bought, think John bought and so on lack an object in essentially the same way that bought alone does, and an operator with category Q/(S/NP) forms a question from an object-lacking sentence. (We are glossing over many details here, including type-raising rules (labeled with “>T”) and generalized composition; see e.g. Steedman 1996 for a more complete presentation.)

(25)

This mechanism generalizes to examples like (3) involving arbitrary numbers of wh-phrases. In (26), we see that the syntactic category (S/NP)/NP associated with Ivan convinced publish includes two NP arguments corresponding to the two gaps, similarly to the way the corresponding constituent in (21) has a node label with two symbols stored in its stack. The propagation of these unfilled argument positions via function composition therefore constitutes an Ext, ¬Fin strategy similar to LIG.
To conclude this section, then, we should observe that there are two viable derivational strategies for the construction of the structural pattern illustrated in (5). One of these, exemplified by TAG, involves maintaining Fin but sacrificing Ext. Another, exemplified by LIG and CCG, maintains Ext but sacrifices Fin. We turn next to examine another MCS formalism, Minimalist Grammars, which have been argued to provide a formal instantiation of minimalist practice, and explore their ability to produce the structural descriptions under examination.

4. Minimalist Grammars (MGs) vs. “everyday minimalism”

A common assumption is that in the abstract configuration in (27), WH₁ prevents WH₂ from moving. Beyond this, however, versions of “Shortest Move” differ. MGs implement a simple conception of Shortest Move, according to which this configuration dooms the derivation, since WH₂ has been prevented from moving to its closest potential attractor. This constraint prevents unchecked featural requirements from accumulating, as they did in (8), and places a bound on the stored information that can condition derivational operations (Fin). Therefore after a certain number of traces have been generated (or, after a certain number of wh-phrases have been introduced into their base positions) it becomes impossible to retain the corresponding requirements for wh-phrase surface positions. Given that they also abide by Ext, MGs will therefore be unable to generate the Bulgarian pattern of tree structures that we have focused on here, for the same reason that LIG with a bound on the stack could not.

(27)
Alternative versions of Shortest Move say that WH₁ can move, which frees up WH₂ for subsequent attractors (e.g. Richards 1997). This involves unbounded storage (¬Fin), because there is no limit to how many elements might be waiting to be “freed up” in this manner; the situation remains equivalent in relevant respects to (8). This Ext-satisfying approach to generating the pattern in (5) is therefore more analogous to LIG than to MG.

The problem posed by the pattern in (5) for MGs is superficially similar to another challenge for MGs that has often been pointed out: cases of multiple wh-fronting such as (28), where multiple wh-phrases appear at the left edge of a single clause, likewise seem to conflict with MGs’ simple Shortest Move constraint.

![Example sentence](28)

Koj₁ kúde₂ misliš [ če Boris iska [ da kažeš [ če šte otide t₁ t₂ ] ] ] ?
Who where you think that Boris wants to you say that will go
Who do you think Boris wants you to say will go where?

Gärtner and Michaelis (2010) propose an amendment to the MG formalism which allows sentences such as (28) to be derived in a way that has antecedents in the mainstream syntax literature, while still maintaining both Fin and Ext. We will show below, however, that this amendment does not solve the problems posed by the pattern introduced in (3) and (4) where fronted wh-phrases occur in separate clauses.

5. Non-context-free stringsets

The argument we have made so far has been based on the assumption that the appropriate hierarchical structures for the crucial Bulgarian sentences have the basic form of those in (7) (rather than those in (6), for example). While there is nothing obviously controversial about this assumption from a linguistic perspective, it is worth considering the status of those structures more explicitly; this will allow us to pinpoint more precisely the important differences between the formalisms under discussion.

5.1 Toy Bulgarian

Recall that without the assumption that the structure for sentences like (3) and (4) should include a constituent that contains all of the traces and none of the wh-phrases, there is nothing of unusual formal complexity about these sentences: they exhibit the abstract structure whⁿ tⁿ, which can straightforwardly be generated by a context-free (string) grammar (CFG). But to do so, a CFG would need to assign constituent structures like those in (29) (repeated from (6) above).
There are many different ways one might argue for the structures we have assumed above rather than these, but a simple one would be by demonstrating the need for independent nesting structure in the portion of these sentences that excludes the wh-phrases. For example, if the sentence in (4) above can be “grown” in a way that adds some elements X and Y according to the pattern illustrated in (30), then a natural analysis would posit a constituent Z that can be self-embedded with an X to its left and a Y to its right, as in (31).

The idea is that such a pattern might lead us to use hierarchical structure to enforce the appropriate X-Y dependencies, in a way that forces us to use some other mechanism to enforce the wh-trace dependencies “vertically”. This assumption that the wh-phrases and traces are arranged vertically was the key point in our earlier discussion in terms of derived structures, and the X-Y dependencies can be thought of as a string-language footprint that induces this structural property. In particular, the two intermingled sets of dependencies correspond to the abstract
string pattern $a^i b^i c^i d^i$, which is not context-free. This observation will allow us to compare the distinctive super-context-free mechanisms of the different families of mildly context-sensitive formalisms (LIG and TAG on the one hand, MG on the other).

We’ll present here a TAG fragment for a fictional language we’ll call “Toy Bulgarian”, which exhibits X-Y dependencies that surround the positions of the wh-traces (but not the wh-phrases) in exactly the manner indicated above. Specifically, Toy Bulgarian will include strings such as those in (32), where each extracted who is the subject of a corresponding said, as models of the Bulgarian pattern introduced in (3) and (4). (The SG subscripts provide a reminder of this dependency, and will become more relevant below.) Toy Bulgarian will also allow the subconstituents from which occurrences of who have been extracted to be (repeatedly) embedded inside additional NP will forget ... tomorrow clauses, playing the role of the X ... Y embeddings in (31), and this will produce the strings in (33). We tentatively assume that appropriate non-fictional analogs of the patterns exhibited by Toy Bulgarian can be found, but we leave the empirical verification of this for future work.

(32)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a. who$<em>{SG}$ NP knew said$</em>{SG}$ it</td>
<td></td>
</tr>
<tr>
<td>b. who$<em>{SG}$ NP knew who$</em>{SG}$ NP knew said$<em>{SG}$ said$</em>{SG}$ it</td>
<td></td>
</tr>
<tr>
<td>c. who$<em>{SG}$ NP knew who$</em>{SG}$ NP knew who$<em>{SG}$ NP knew said$</em>{SG}$ said$<em>{SG}$ said$</em>{SG}$ it</td>
<td></td>
</tr>
</tbody>
</table>

(33)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a. who$<em>{SG}$ NP knew NP will forget said$</em>{SG}$ it tomorrow</td>
<td></td>
</tr>
<tr>
<td>b. who$<em>{SG}$ NP knew NP will forget NP will forget said$</em>{SG}$ it tomorrow tomorrow</td>
<td></td>
</tr>
</tbody>
</table>

The TAG fragment for Toy Bulgarian will be introduced in three parts. The first part is shown in (34).

(34)

The leftmost auxiliary tree, labeled $\beta_1$ (following the convention from the TAG literature of labeling auxiliary trees with a subscripted $\beta$), introduces the dependencies that we have so far described as holding between a wh-phrase and a trace.$^6$ These are now realized (in the string language) as dependencies between the wh-phrase who and a verb said lacking a local overt subject. Note that the foot node of $\beta_1$ is annotated with NA, indicating a null adjoining constraint.

$^6$ The $t$ node in the auxiliary tree $\beta_2$, representing a trace, is formally identical to an epsilon node; we notate them differently only for readability.
This means that adjoining cannot apply at this node. In contrast, the root node is annotated with SA followed by a set containing other auxiliary trees. This selective adjoining constraint restricts the auxiliary trees that can adjoin at this node to those listed in the set (in this case, an auxiliary tree we introduce below). The middle auxiliary tree, labeled $\beta_2$, introduces a dependency between *will forget* and *tomorrow*, corresponding to the separate nesting dependencies indicated abstractly with X and Y in (30). As in $\beta_1$ and following the general TAG convention, the foot node in this tree is labeled with NA, prohibiting adjoining at the foot. The root node is also labeled with an SA constraint. Finally, the rightmost elementary tree, $\alpha_1$, is an initial tree corresponding to a pronominal clausal complement.

A derivation of the string in (35), involving two *who-said* dependencies, is shown in (36).

(35) who$_{SG}$ who$_{SG}$ said$_{SG}$ said$_{SG}$ it
(36) 

On the right of (36) is a TAG derivation tree indicating the sequence of derivational steps that have been employed to derive the structure on the left. Each node in a derivation tree corresponds to an instance of an elementary tree in the grammar. When one elementary tree $T_1$ is the daughter of another one $T_2$ in a derivation tree, this indicates that $T_1$ has been combined into $T_2$. In this derivation, which is most easily interpreted in a bottom-up fashion, one instance of $\beta_1$ is adjoined into another at the internal CP node, yielding a derived CP-recursive auxiliary tree. To complete the derivation, this derived auxiliary tree is adjoining into $\alpha_1$ at the root. Strings containing unboundedly many such dependencies can be derived in an analogous fashion, by adjoining in additional instances of $\beta_1$ before finally adjoining the result into $\alpha_1$.

---

7 Such daughterhood relations are also typically annotated with the identity of the TAG combinatory operation (adjoining or substitution) and the locus of the operation. Because we use only adjoining here, we omit the distinction between substitution and adjoining. We note the locus of adjoining in the text.
To wrap the sequence of X-Y dependencies around the locus of the wh-gap(s), we make use of the auxiliary tree $\beta_2$. We can adjoin copies of this auxiliary tree into itself at the root, to derive the complex auxiliary tree shown in (37).

(37)

```
CP
   NP will forget CP tomorrow
      NP will forget CP tomorrow
         NP will forget CP tomorrow
```

Adjoining this into $\beta_1$ before continuing as in (36), we will derive:

(38) $\text{who}_{\text{SG}} \text{ who}_{\text{SG}} [\text{NP will forget} [\text{NP will forget} [\text{NP will forget} [t \text{ said}_{\text{SG}} [t \text{ said}_{\text{SG}} \text{ it } ] ] \text{ tomorrow } ] \text{ tomorrow } ] \text{ tomorrow } ]$

More generally, the stringset generated by just this first portion of the fragment is (39).

(39) $L_1 = \text{who}_{\text{SG}}^i (\text{NP will forget})^i \text{ said}_{\text{SG}}^j \text{ it tomorrow}^j$

Notice that both the $\text{who-said}$ and the $\text{will-tomorrow}$ dependencies here are nested: the last $\text{who}$ is introduced into the derivation through the same elementary tree that introduced the first $\text{said}$. Such a dependency is not reflected in the string itself, since all instances of these elements are identical. In order to render these dependencies “visible” to the string language, we can include two new elementary trees that act as counterparts to $\beta_1$ and $\beta_2$: $\beta_3$ is a counterpart to $\beta_1$ that includes plural markings on $\text{who}$ and $\text{said}$ instead of singular markings, and $\beta_4$ is a counterpart to $\beta_2$ that includes $\text{did}$ and $\text{yesterday}$ instead of $\text{will}$ and $\text{tomorrow}$. These are shown in (40), the second of the three parts of our fragment.

(40)

```
$\beta_3$

CP$_{SA} \{\beta_3\}$

$\text{who}_{\text{PL}}$

CP

$\varepsilon$

TP

$\text{said}_{\text{PL}}$

CP$_{NA}$

$\beta_4$

CP$_{SA} \{\beta_2, \beta_4\}$

NP did forget CP$_{NA}$ yesterday
```

Adding these to our grammar allows us to generate strings with dependencies like this (omitting irrelevant elements, specifically occurrences of $\text{forget}$ and $\text{it}$):
In this style of derivation, where all adjoining of $\beta_1$ and $\beta_3$ takes place at the internal CP node of each tree, all occurrences of ‘who.SG’ and ‘who.PL’ will appear adjacent to each other, intuitively all at the left edge of the same clause, in the manner usually associated with multiple questions. To allow the pattern introduced in (3) and (4) that we have focused on, where the wh-phrases occupy the left edges of distinct clauses, we include the auxiliary tree $\beta_5$ shown in (42), which can be adjoined at the root of $\beta_1$ or $\beta_3$. (This is the third and final part of our fragment.)

For example, if we adjoin an instance of $\beta_5$ into $\beta_1$ before continuing as in (36), we will derive the following structure, where the two occurrences of who are no longer adjacent.
This separation of the wh-phrases can occur along with the additional nesting dependencies introduced by $\beta_2$ and $\beta_4$, of course. So the completed fragment generates a string language essentially of the form in (44) (erasing the SG/PL distinction, collapsing will and did to aux, and collapsing tomorrow and yesterday to adverb).

(44) \[ L_2 = (\text{who} \ (\text{NP knew})^*)^i \ (\text{NP aux forget})^i \ \text{said}^i \ \text{it adverb}^i \]

The adjunction constraints ensure that things can only be combined in the ways we’ve described here.

5.2 Back to MGs

What we’ve shown so far is that the Toy Bulgarian stringset sits in the TAG/LIG/etc. region of the diagram in (1). Moreover, this TAG generates the Toy Bulgarian sentences with a “sensible” structure that is in line with standard linguistic analyses. Although we have presented the grammar as a TAG, an equivalent LIG can easily be constructed: intuitively, it will use the “normal” center-embedding capabilities of a CFG to mediate the auxiliary-adverb dependencies, and use its stack to track the wh dependencies by remembering the vertical nesting of $\beta_1$ and $\beta_3$. (And a CCG which does something similar could be constructed too.) Nonetheless, regardless of which of these grammar formalisms we invoke, the structure that is derived is in essential respects identical to what is usually assumed for this kind of sentence in the minimalist syntax literature.

As mentioned in the introduction, any advantages of TAG/LIG over MG cannot be in being able to generate the Toy Bulgarian stringset, for it has been shown that the stringsets generated by MGs properly include those generated by TAGs/LIGs (Michaelis 2001). As a result, there must exist an MG that generates the same string set as the TAG we have presented. But this is a non-context-free stringset, and it has also been shown that standard MGs can only generate non-context-free stringsets by using either remnant movement or head movement (Michaelis 2002, Kobele 2010). As a result, every MG that generates the Toy Bulgarian stringset will do so via either remnant movement or head movement; no combination of non-remnant phrasal movements will work. Note, though, that minimalist-style analyses of the original Bulgarian pattern do not invoke these operations, instead favoring non-remnant phrasal movement to account for the displacement of the wh-phrases (i.e., as indicated by the traces in the structure derived in the TAG derivation). Consequently, MGs appear to be unable to express the minimalist-style analysis that we, and the minimalist syntax literature, are taking to be the natural one. This raises the question of the degree to which MGs are a formal instantiation of practice in minimalist syntax.

It is now useful to return to a proposed extension of MGs that we mentioned briefly above. Gärtner and Michaelis (2010) introduce a clustering operation motivated by sentences like (28), repeated here as (45).

(45) \[ \text{Koj}_1 \ \text{kűde}_2 \text{ misliš } \ [ \text{če } \text{ Boris iska } [ \text{da } \text{kažeš } [ \text{če } \text{šte otide } t_1 \ t_2 ] ] ] \ ? \]

Who do you think Boris wants you to say that will go where?

Such a sentence raises the same unbounded-state difficulties for standard MGs that we outlined above in (8). In (46) we illustrate this with a derivation where the two movements occur in a
“tucking-in” order (Richards 1999), but this detail does not affect the important point that at step (b) and (c) all wh-phrases have unchecked features. More generally, an analogous derivation of a sentence that has \( n \) fronted wh-phrases will include a point where \( n \) wh-phrases have unchecked features, so as the pattern extends the required derivational state will increase without bound, contra MGs’ commitment to \textit{Fin}.

\begin{equation}
(46)
\begin{align*}
a. \text{will go where} & \quad \text{unchecked: 1} \\
b. \text{[TP who will go where]} & \quad \text{unchecked: 2} \\
c. \text{[TP you think Boris wants you to say [TP who will go where]]} & \quad \text{unchecked: 2} \\
d. \text{[CP who [TP you think Boris wants you to say [TP t will go there]]]} & \quad \text{unchecked: 1} \\
e. \text{[CP who where [TP you think Boris wants you to say [TP t will go t]]]} & \quad \text{unchecked: 0}
\end{align*}
\end{equation}

Gärtner and Michaelis introduce a derivational operation which groups the as-yet-unmoved wh-phrases into a single cluster. Using this operation, the derivation of (45) instead proceeds as sketched in (47). The innovation is the operation that allows \textit{where} to move and form a cluster with \textit{who} at step (c). Note that this is not movement of both wh-phrases to a new position at the left edge of a clause; the two wh-phrases are clustered to form a single constituent that occupies the subject position where \textit{who} is first merged. In the final step of the derivation, a single wh-movement step displaces this cluster.

\begin{equation}
(47)
\begin{align*}
a. \text{will go where} & \quad \text{unchecked: 1} \\
b. \text{[TP who will go where]} & \quad \text{unchecked: 2} \\
c. \text{[TP who where will go t]} & \quad \text{unchecked: 1} \\
d. \text{[TP you think Boris wants you to say [TP who where will go t]]} & \quad \text{unchecked: 1} \\
e. \text{[CP who where [TP you think Boris wants you to say [TP t will go t]]]} & \quad \text{unchecked: 0}
\end{align*}
\end{equation}

The crucial point is that as soon as this cluster is formed in step (c), it can be treated as a single element with unsatisfied movement-triggering requirements. After step (c), the derivation does not need to “know” whether the as-yet-unmoved element in that subject position is a cluster or a single wh-phrase; there is no need to track the distinction between [TP who where will go t] as in (47) and alternatives such as [TP who will go] or [TP who will go there], for example. The counter of unchecked requirements therefore decreases to 1 as soon as the clustering occurs in step (c). The important consequence of this is that as the pattern is extended to analogous sentences with more than two wh-phrases, the derivational state remains bounded: there are never more than two elements with unchecked featural requirements. If we replace \textit{Boris} in (47) with \textit{who}, for example, the resulting derivation involving three wh-phrases in total will proceed as sketched in (48).

\begin{equation}
(48)
\begin{align*}
a. \text{will go where} & \quad \text{unchecked: 1} \\
b. \text{[TP who will go where]} & \quad \text{unchecked: 2} \\
c. \text{[TP who where will go t]} & \quad \text{unchecked: 1} \\
d. \text{[TP you think who wants you to say [TP who where will go t]]} & \quad \text{unchecked: 2} \\
e. \text{[TP you think who who where wants you to say [TP t will go t]]} & \quad \text{unchecked: 1} \\
f. \text{[CP who who where [TP you think t wants you to say [TP t will go t]]]} & \quad \text{unchecked: 0}
\end{align*}
\end{equation}
The clustering operation therefore prevents unchecked featural requirements from accumulating as the number of moving phrases grows. This extension of the MG formalism therefore allows for sentences like (45) to be derived in a way that maintains both Ext and Fin, and looks similar to proposals in the “informal” syntax literature for languages like Bulgarian (Rudin 1988, Grewendorf 2001).

How might MGs enriched with this clustering operation handle the patterns in our Toy Bulgarian fragment discussed above? Corresponding to the TAG derivation illustrated in (36), Gärtner and Michaelis’s MGs will make available a derivation that produces the structure in (49). The one notable difference (aside from the issue of binary versus ternary branching; only TAG allows ternary branching) is the arrangement of the two wh-phrases into a single constituent in (49): the first movement step here merges the lower wh-phrase with the higher one in the specifier of TP position. This cluster is then moved to the specifier of CP in the final derivational step. In (36), in contrast, the wh-phrases are separate, independently displaced specifiers of the matrix CP. But we treat both of these options as linguistically reasonable.

---

8 Graf and Kostyszyn (2021) make a related proposal to accommodate cases like (45). Graf and Kostyszyn observe that to bring these examples into line with Fin and Ext, what matters is that we somehow collapse the effects of the multiple wh-phrases on derivational state; for Gärtner and Michaelis this collapsing of derivational state arises via the formation of a cluster at the level of derived structure, but Graf and Kostyszyn characterize the crucial collapsing of state via an adjustment at the level of the feature calculus (specifically, the introduction of persistent licensor features) which abstracts away from specific choices of derived structure. For our purposes here, what is important is that Graf and Kostyszyn's novel feature-checking mechanism, like Gärtner and Michaelis's clusters, allow for the normally one-to-one relationship between attractors and attractees to become one-to-many. As we emphasize in the remainder of this section, the derivations that provide the most significant challenge for MGs' conjunction of Ext and Fin are those involving many-to-many relationships. For derivations involving many-to-many dependencies, the unbounded-state problem for MGs discussed in section 3 cannot be circumvented by collapsing the effects of multiple yet-to-be-moving phrases.
MGs with clustering will also provide reasonable derivations of sentences of the sort shown in (38), where additional NP *will forget ... tomorrow* clauses surround the *said* clauses. As shown in (50), this simply requires additional standard merge steps, which will occur after all wh-phrases have been clustered in the subject position of the highest *said* (at step (b)), but before this cluster is moved into its surface position (at step (d)).
This kind of derivation is therefore sufficient to produce the intermediate stringset $L_1$ from (39) above.

$$L_1 = \text{who}_{\text{SG}}^i (\text{NP will forget})^i \text{said}_{\text{SG}}^i \text{it tomorrow}^i$$

Since $L_1$ is not context-free, this shows that clustering can serve as an alternative path to non-context-freeness for MGs; recall that in standard MGs, non-context-freeness necessitates either remnant movement or head movement.
Now, from a formal perspective, the full stringset $L_2$ of our Toy Bulgarian fragment (repeated here from (44)) is not substantially different from $L_1$. The only difference is the optional occurrences of $NP$ knew after each who.

$L_2 = (\text{who} \ (NP \ \text{knew})^*) \ (NP \ \text{aux} \ \text{forget})^j \ \text{said}^j \ \text{it adverb}^j$

So the same formal conclusions about the ways in which this will be derivable apply: if an MG derives $L_2$ without using remnant movement or head movement, the only remaining option will be wh-clustering. But in $L_2$, the unboundedly many wh-elements themselves need not be linearly contiguous. Deriving such examples with a wh-clustering analysis would therefore require that the additional $NP \ \text{knew}$ be part of the cluster into which the wh-phrases have been collapsed.\footnote{Note also that no operation excorporating a wh-phrase from a cluster is possible: this would require the recovery of information that had been abandoned when the cluster was formed in order to satisfy Fin.} An outline of one way that this might work is shown in (51).
There are many ways to spell out the details of how the \textit{NP knew} material is introduced. Notice, however, that because \textit{NP knew} is linearly in between the two wh-phrases, there is no alternative to (51) where the final cluster arises from combining \textit{NP knew} with a previously-formed (and sensible-looking) cluster that contains only the two wh-phrases. This means that \textit{NP knew} is necessarily introduced into the derivation before the surrounding \textit{will ... tomorrow} elements are.

To recap: in the case of \(L_1\), neither remnant movement nor head movement provides a linguistically natural derivation, while the wh-clustering option does. This forces MGs to use structures like (49), which differ from the TAG/LIG approach, but still provides a linguistically plausible alternative. For \(L_2\) on the other hand, the same reasoning leads to the conclusion that MGs are unable to derive structures where the wh-elements are separated as in (37), and must instead use structures like (51), which do not seem to be a linguistically plausible alternative. Such a structure has no antecedent in the informal syntax literature, to our knowledge.

It is interesting to contrast the construction we have been discussing with another non-context-free pattern found in natural language: the famous Dutch and Swiss German crossing-dependencies pattern (Shieber 1985). For this case, the syntax literature contains numerous proposals that invoke the mechanisms of head movement (Evers 1975) and remnant movement (Haegeman and van Riemsdijk 1986, Koopman and Szabolcsi 2000) in order to account for this pattern, so that both lines of analysis are compatible with MGs.\footnote{See however Kroch and Santorini (1991) for arguments that neither of these approaches is sufficient to account for the full range of word order patterns found across the West Germanic languages. The relationship between Kroch and Santorini's analysis of the Germanic patterns and the head movement analyses of such patterns, mirrors the relationship between the TAG and MG strategies for generating the Toy Bulgarian stringset.} Interestingly, these proposals were made independently of concerns of generative capacity, and before the central role of exactly these mechanisms in generating non-context-free languages was identified.

Such analytic heterogeneity highlights a distinctive property of the (Toy) Bulgarian pattern we have discussed here: it is a weakly non-context-free string pattern whose structural analysis is much less controversial than the Germanic crossing dependencies. It therefore provides a case where established linguistic assumptions can be much more meaningfully brought to bear on choices between the differing super-context-free mechanisms of various MCS formalisms.

\section*{6. Conclusion}

The central point of our discussion has been that the pattern of extractions in wh-questions in languages like Bulgarian involving multiple wh-movement dependencies — if it extends without bound as we have assumed — is incompatible with systems of derivation that enforce the conjunction of \texttt{Ext} and \texttt{Fin}. As we have seen, MCS formalisms can be classified according to these properties, and the Bulgarian pattern therefore provides an argument in favor of formalisms that abandon \texttt{Ext}, such as TAG, or \texttt{Fin}, such as LIG or CCG.

\vspace{1em}(52)
The argument against standard variants of MGs is particularly striking given its greater weak generative capacity. Moreover, the inability of MGs to capture what we take to be the standard analysis of the Bulgarian phenomenon that we began with raises questions about the validity of taking MG to be the MCS formalism that best aligns with standard minimalist practice, as compared to TAG, CCG or LIG.

We have left unexplored the question of the nature of the relation between the fronted wh-phrases and the associated traces. In fact, the formalisms we have been considering do make predictions about how these dependencies will be configured. LIG’s stack-structured memory will necessarily generate wh-trace dependencies that are structurally nested, as does TAG’s adjoining operation. On the one hand, this result is a positive one, in accordance with the empirically-motivated Path Containment Condition (Pesetsky 1982). However, we recognize that the facts might be more complex in these cases (Richards 1997). We note, therefore, that the expressive capacity for structural description shared by LIG and TAG is just the first step on a hierarchy that can be defined either via different relaxations of Fin (Weir 1992) or different relaxations of Ext (Rogers 2003). The different points on this hierarchy will make distinct predictions about the possible structural patterns of wh-trace configurations. We leave the investigation of these for future work.

References


Non-standard variants of MGs that depart from the conjunction of Ext and Fin include Graf (2012) (which rejects Ext) and Gärtner and Michaelis (2005) (which rejects Fin). Kobele and Michaelis (2005) showed that the latter has the power of a Turing machine, and is therefore not MCS. We leave for the future an examination of the properties of Graf’s proposed variant.


Mönnich, Uwe. 2007. “Minimalist Syntax, Multiple Regular Tree Grammars and Direction Preserving Tree Transductions”. In *Proceedings of Model-Theoretic Syntax at 10*, pp.83--88.


