revised version in C. Boeckx, ed. (2010) *Oxford Handbook of Linguistic Minimalism*, pp.616-641.

Edward P. Stabler

Computational perspectives on minimalism

While research in 'principles and parameters' tradition [18] can be regarded as attributing as much as possible to universal grammar (UG) in order to understand how language acquisition is possible, Chomsky characterizes the 'minimalist program' as an effort to attribute as little as possible to UG while still accounting for the apparent diversity of human languages [23, p.4]. Of course, these two research strategies aim to be compatible, and ultimately should converge. Several of Chomsky's own early contributions to the minimalist program have been fundamental and simple enough to allow easy mathematical and computational study. Among these contributions are (1) the characterization of 'bare phrase structure,' and (2) the definition of a structure building operation *merge* which applies freely to lexical material, with constraints that 'filter' the results only at the PF and LF interfaces. The first studies inspired by (1) and (2) are 'stripped down' to such a degree that they may seem unrelated to minimalist proposals, but in this paper, we show how some easy steps begin to bridge the gap.

This paper briefly surveys some proposals about (3) syntactic features that license structure building, (4) 'locality,' the domain over which structure building functions operate, (5) 'linearization', determining (in part) the order of pronounced forms, and (6) the proposal that merge sometimes involves 'copies.' Two very surprising, overarching results emerge. First, seemingly diverse proposals are revealed to be remarkably similar, often defining identical languages, with recursive mechanisms that are similar (Thms. 2-5, below). As noted by Joshi [47] and others, this remarkable convergence extends across linguistic traditions and even to mathematical work that started with very different assumptions and goals (Thm. 1, below). Second, all the mechanisms reviewed here define sets of structures with nice computational properties; they all define 'abstract families of languages' (AFLs) that are efficiently recognizable. This raises an old puzzle: why would human languages have properties that guarantee the existence of parsing methods that correctly and efficiently identify all and only the well-formed sentences, when humans apparently do not use methods of that kind? A speculation – perhaps supported by the whole cluster of AFL properties – is that this

UCLA Department of Linguistics

might facilitate learning, by facilitating the calculation of how lexical properties should be adjusted in light of input available to the learner.

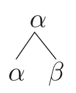
One methodological point should be noted immediately. One way to study generative mechanisms is by considering what they generate, what structures and what 'sentences'. Often this is the first, easiest thing to assess. This does *not* indicate that the linguists' task is to provide grammars generating all and only some set of pronounced sequences that are judged 'grammatical'. For one thing, syntactic principles (at least to a good first approximation) do not depend on phonetic properties, and so in our formal studies, the alphabet of pronounced complexes gets scant attention – in a certain sense, we do not care what the sequences are, but only what kinds of patterns they exhibit. And morphophonology also intervenes to obscure the sequences of syntactic heads in ways that are not well understood. But most importantly, in studying syntax, we abstract away from many 'non-syntactic' influences on language, factors which are not known in advance. Spoken phonetic sequences, intuitive judgments about 'sentences', and corpus studies can provide our evidence, in part, but we do not know *a priori*, in advance of the science, what should count as a 'sentence', or which properties of language can be explained in which ways. This methodological stance is familiar from [16, Chapter 1, Section 1] and many other places.

1. Bare phrase structure

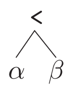
Following Muysken [71], Chomsky [20, pp.242-243] suggests that the important insight of X-bar syntax is a relational one: a head X determines certain relevant properties of the phrase XP it is the head of. This idea, sometimes called 'endocentricity,' gets its content with a specification of what comprises a phrase XP and which properties are relevant. Which properties of the complex XP are visible later in the derivation, and which are determined by the head? These properties are sometimes called the 'label' of the complex: Chomsky [23, p.17] says "all operations are driven by labels." If the labels encode what is visible to syntactic operations, they must encode at least those properties of the head that have an influence on later derivational steps, and any properties of other elements that can enter into other syntactic relations ('valuation', 'licensing') elsewhere in the derivation. Chomsky [20, p.245] suggests that merging constituents α and β yields a set $\{\alpha, \beta\}$ together with label α,

(1) $\{\alpha, \{\alpha, \beta\}\}$,

a complex that could be also be regarded as an unordered tree:[1]

(2) 

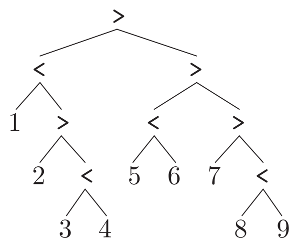
The set notation and the tree both represent α twice, but that is not necessary.[2] A slight variant of this notation from [91] represents α just once, with a tree notation in which internal nodes are labeled with order symbols (< or >) 'pointing' towards the head:

(3) 

We could label the leaves of these trees with lexical items, but we can further reduce clutter in the representation by assuming that all and only the syntactic features of heads are visible, where these features are ‘erased’ once they cease to play a role. In that case, the leaves will either be lexical items or simpler structures.

Some research in the minimalist program also assumes that when elements are merged, their linear order (and possibly even the question of whether they will be pronounced) may not be determined until later in the derivation. We will postpone consideration of this abstraction, and tentatively assume that linear order can be determined locally by the nature of the elements merged, forming complexes in which linear order can already be determined. So we will use the notation of (3), departing from (2) not only with the indication of headedness but also with the assumption that *the tree is linearly ordered*. Some adjustments to this tentative assumption, and other ideas about ‘linearization’, are briefly considered in §5 below.

With these assumptions, consider a tree like this:

(4) 

The head of this whole tree is the node labeled 8. Every tree is regarded as a subtree of itself, and the leaves are subtrees too. Let’s say that a subtree is a ‘maximal phrase’, or a ’maximal projection’, or simply ‘maximal’, if it is not properly included in any larger subtree that has the same head. The ‘minimal’ elements in each tree are the leaves. Then the subtree of (4) with leaves 234 is maximal, while the subtree containing only 34 is neither maximal nor minimal, and the subtree containing only the leaf 4 is both maximal and minimal. Furthermore, with the assumption that the tree is linearly ordered, the heads are pronounced in the order 123456789.

Considering the lexical items more carefully, let’s assume that they have semantic and phonetic features, which are distinct from the formal syntactic features. We will put the non-syntactic features first (usually using the conventional spelling of a word to indicate what is intended), followed by a double colon ::, followed by a sequence of syntactic features.

$$\text{Phon} :: \text{feature1 feature2}\ldots\text{featureN}.$$

We use the double colon :: in lexical items, but for reasons mentioned in Appendix 1, in derived structures a colon : will separate phonetic from syntactic features – in the body of this paper this distinction will be respected but will not matter. And here we assume that syntactic features are ordered sequentially, but an alternative is considered in §3, below.

Tentatively, let's distinguish four kinds of syntactic features: in addition to the usual 'categorial' features N, V, A, P, C, T, D,..., let's assume that a head which wants to select a phrase of category X has a features =X. So then we have the 'selector' features =N, =V, =A, =P, =C, =T, =D.... Ultimately, of course, we would like to understand the nature of these features – why some verbs select DP arguments, for example, but for the moment we simply give such a verb a feature =D.[3] Some other properties may require or allow a licensing relationship of a different kind, so we will have 'licensee' or 'goal' features -wh, -focus, -case,.... We assume, initially, that these features are all distinct; for example, no licensee feature is also a category. The heads that can license phrases with such requirements will be called 'licensors' or 'probes' and will have features +wh, +focus, +case,.... So a simplistic lexicon might have 4 items like this:

Marie	::	D
who	::	D -wh
praises	::	=D =D V
ϵ	::	=T +wh C.

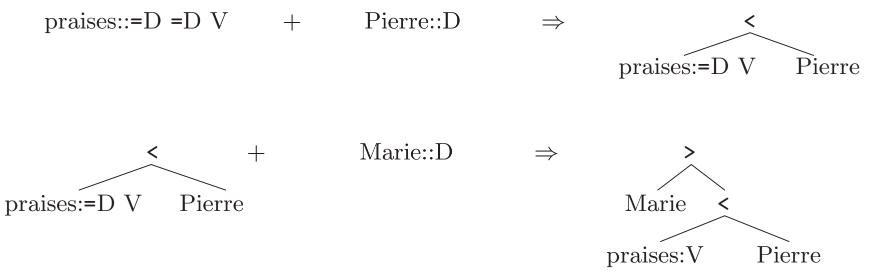
A **minimalist grammar (MG)** is simply a finite lexicon like this. That is, a minimalist grammar G is a lexicon

(5) $G \subset \text{Phon} \times \text{Features}^*$, a finite set,

where Features* is the set of finite sequences of syntactic features, and where the elements of the lexicon are combined by the merge operation which is defined in §2, just below. Later sections consider some variations in both the feature system and in the the merge function.

2. Merge: First version

We will regard the lexicon as providing the labels for 1-node trees, so that merge can be regarded as a function that applies either to pairs of trees ('external merge', em) or to single trees ('internal merge', im). Since we are assuming that the derived structures specify linear order (an option we reassess in §5 below), em is specified with two cases as well. When a lexical selector combines with a first element, that element is attached on the right and is called the complement. When a derived expression selects another element, that element is attached on the left and is called a specifier. (Some linguists have proposed that each category can have at most one specifier, but these first definitions will not impose that bound, allowing any number of specifiers.) Furthermore, we assume that the selector features =X and X must be the first features of the heads of the arguments, and that they are both erased by merge. For example, applying merge to the pairs of structures shown on the left, we obtain the derived structures on the right:



Let's write $t[\alpha]$ when the head of tree has a sequence of syntactic features whose first element is α. Given a structure $t[\alpha]$, let t denote the result of (i) erasing feature α and (ii) if the head of $t[f]$ has a double colon ::, changing it to a colon. And for any tree t, let $|t|$ be the number of nodes in t. Then the function em is given by

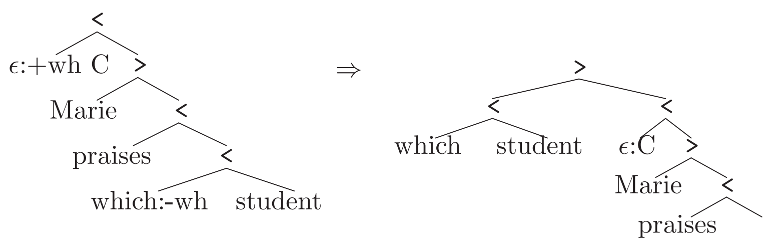
(6) $$\mathbf{em}(t_1[\texttt{=x}], t_2[\texttt{x}]) = \begin{cases} \overset{<}{t_1 \quad t_2} & \text{if } |t_1| = 1 \\ \overset{>}{t_2 \quad t_1} & \text{otherwise} \end{cases}$$

To reduce notational clutter, we write a leaf node label *word*:ϵ simply as *word*, as in the *Marie praises Pierre* examples above. And leaf nodes with no features at all, $\epsilon : \epsilon$, are usually written just ϵ, or as nodes with no label. In particular, the tree consisting of one node and no (phonetic, semantic or syntactic) features is sometimes called ϵ.

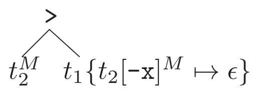
Internal merge applies to a single structure $t[\texttt{+x}]$ only if it satisfies this strong version of the 'shortest move constraint':

(7) SMC: Exactly one head in the tree has -x as its first feature.

In that case, im moves the maximal projection of the -x head to specifier position, leaving an empty subtree ϵ behind. (§6.5 considers the idea that im involves a copy, leaving the original -x phrase in its original position. Covert movements that leave the phonetic material behind are briefly discussed in §6.2 below.) So for example,



To define this operation, given any tree t, let $t\{t_1 \mapsto t_2\}$ be the result of replacing subtree t_1 by t_2 in t, and given any subtree (possibly a leaf) t, let t^M be the maximal projection of the head of t. Then im applies to a tree $t_1[+x]$ containing subtree $t_2[\texttt{-x}]$, by deleting the +x feature to obtain t_1, removing the maximal projection of the -x head to obtain $t_1\{t_2[\texttt{-x}]^M \mapsto \epsilon\}$, and finally adding t_2^M as a specifier. In sum,

(8) $\mathbf{im}(t_1[\texttt{+x}])$ $=$ 

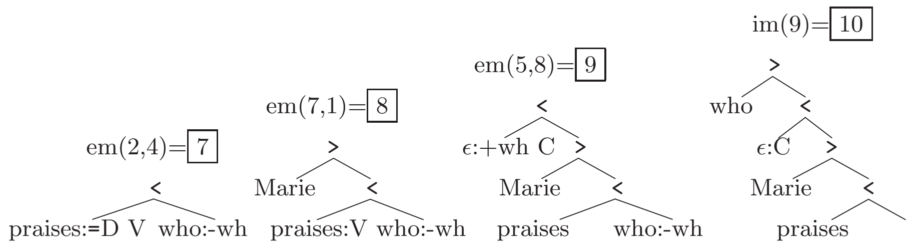
if SMC.

Since em and im apply in different cases, their union is also a function, which we will call **merge**.

Example G1. Consider the grammar G1 given by these 6 lexical items, numbered for easy reference:

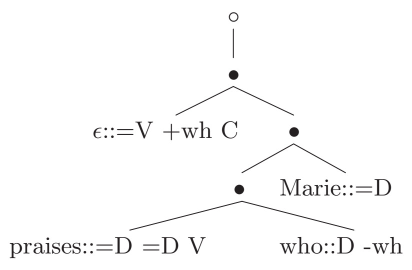
$\boxed{0}$	Pierre::D	who::D -wh	$\boxed{4}$
$\boxed{1}$	Marie::D	ϵ::=V +wh C	$\boxed{5}$
$\boxed{2}$	praises::=D =D V	knows::=C =D V	$\boxed{6}$
$\boxed{3}$	ϵ::=V C		

With this grammar G, we can apply merge (em and im) as follows, for example:



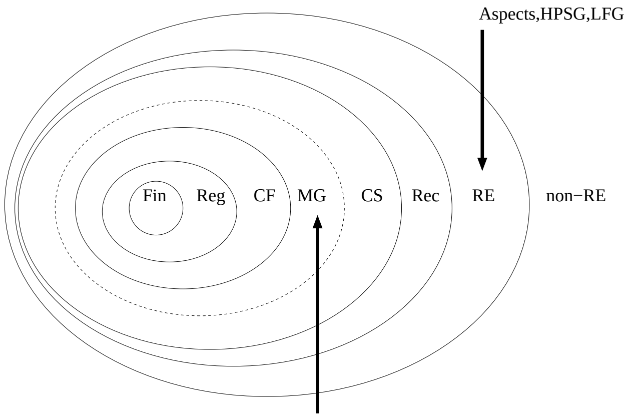
For any MG G, let the set **structures(G)** includes all and only the structures that can be derived from the lexical items of G by applying merge in all possible ways. Let a **completed** structure is one in which (i) there is exactly 1 syntactic feature, (ii) that feature is the 'root' or 'start' category, and (iii) that feature is at the head of the tree. And let the language **L(G)** be the set of phonetic sequences at leaves of completed structures of G.

The derivation tree of the 4 step derivation of our example can be drawn like this:



Here we use • to represent em, and ∘ to represent im. Since these functions apply unambiguously, the derived structure obtained at each internal node of this derivation tree is completely determined. So if C is the 'start' category of our example grammar G, then this derivation shows that the sentence *who Marie praises* $\in L(G1)$. Notice that the derivation tree is not isomorphic to the the derived tree, numbered $\boxed{10}$ just above.

Minimalist grammars (MGs), as defined here by (5), (6) and (8), have been studied rather carefully. It has been demonstrated that the class of languages definable by minimalist grammars is exactly the class definable by multiple context free grammars (MCFGs), linear context free rewrite systems (LCFRSs), and other formalisms [62,64,66,41]. MGs contrast in this respect with some other much more powerful grammatical formalisms (notably, the 'Aspects' grammar studied by Peters and Ritchie [76], and HPSG and LFG [5,46,101]):



The MG definable languages include all the finite (Fin), regular (Reg), and context free languages (CF), and are properly included in the context sensitive (CS), recursive (Rec), and recursively enumerable languages (RE). Languages definable by tree adjoining grammar (TAG) and by a certain categorial combinatory grammar (CCG) were shown by Vijay Shanker and Weir to be sandwiched inside the MG class [103].[4] With all these results,

Theorem 1. $CF \subset \boxed{TAG \equiv CCG} \subset \boxed{MCFG \equiv LCFRS \equiv MG} \subset CS.$

When two grammar formalisms are shown to be equivalent ($\equiv$) in the sense that they define exactly the same languages, the equivalence is often said to be 'weak' and possibly of little interest to linguists, since we are interested in the structures humans recognize, not in arbitrary ways of defining identical sets of strings. But the weak equivalence results of Theorem 1 *are* interesting. For one thing, the equivalences are established by providing recipes for translating one kind of grammar into another, and those recipes provide insightful comparisons of the recursive mechanisms of the respective grammars. Furthermore, when a grammar formalism is shown equivalent to another one that is already well studied, many new facts about the new formalism may come to light; this in fact happened in the case of MGs.

One key insight behind Theorem 1 can be expressed as follows [63, 66, 62, for full details]. For any MG G, let's say that a derived tree in structures(G) is **useful** or **relevant** if and only if it is used in a derivation of a completed structure. That is, completed structures are useful, and so are all the structures involved in their derivations. Let useful(G) be the set of useful elements of structures(G). Then it is easy to see that every minimalist grammar G has the following property:

(9) **(Finite partition)** $\exists n \geq 0$, $\forall t \in$ useful(G), the number of heads in t with syntactic features is less than n. So every useful structure can be classified according to which features these heads have, providing a finite partition of useful(G).

MGs have this property because, first, there can only be a finite number of lexical items, and so there can only be a finite number of licensee features. Second, each head in a useful tree will have some suffix of the syntactic feature sequences in the lexicon, since syntactic features are checked and erased in order, from the front. And third, by the SMC, no useful tree can have two heads beginning with the same licensee. So there can only be finitely many heads with non-empty sequences of syntactic features. Classifying each useful tree according to these feature sequences, in a relevant sense, completely determines syntactic properties of that tree.

Michaelis [63, §5.2] shows that MG languages have a large set of nice closure properties that make the class a 'substitution-closed full abstract family of languages' (AFL) in the standard sense introduced in [35] and discussed in [59, §3].[5] This means, for example, that it is easy to represent the intersections of minimalist languages and the results of certain kinds of substitutions and other operations. Many standard parsing methods and probabilistic models depend implicitly on AFL properties [39, 72, 36].

It is also known that the MG definable languages are efficiently parsable [90], and that standard parsing methods can be adapted for them [42].[6] In CKY and Earley parsing models, the operations of the grammar (em and im) are realized quite directly by adding, roughly, only bookkeeping operations to avoid unnecessary steps. For any minimalist grammar G, these parsing models are guaranteed to accept all and only the elements of L(G). (In cases of non-sentences these parsing methods will not 'succeed', but will often detect non-trivial subconstituents, a presumably useful input to repair or learning strategies.) Humans, on the other hand, in the recognition of fluent speech, seem to use parsing methods that fail on certain kinds of fully grammatical structures; among the best-known examples are garden paths like *the horse raced past the barn fell* [77], and highly ambiguous strings like *police police police police police police* [4, §3.4]. It is an open question whether the human failures can be attributed to externally imposed limitations on mechanisms that can otherwise handle all constructions definable with merge (cf. the chapter on processing in this volume, and the discussion of grammar-performance relations in [6]).

Earlier formal studies of government-binding (GB) theory [86,87,55] concluded that it was a context free grammar notation, up to indexing.[7] But those GB grammars required that each moved constituent c-command its trace, blocking 'remnant movement'. That requirement, sometimes called a 'proper binding condition' (PBC), is now generally regarded as too stringent [2,9,44,70,69,53,48], and is not imposed here. Familiar remnant movement analyses include structures like these:

(10) a. $[_{AP_2}$How likely $[t_1$ to win$]]$ is$_3$ John$_1$ t_3 t_2?

b. John $[_{VP_2}$reads $t_1]$ [no novels]$_1$ t_2.

c. $[_{VP_2}$ t_1 Gelesen$]$ hat [das Buch]$_1$ [keiner t_2].
read has the book noone

Notice that without the PBC, many of the common, simplistic assumptions about processing models will not work: processors cannot work left-to-right by putting moved elements into a memory store and then watching for the corresponding gap, since human languages not only allow constructions in which gaps precede the moved elements, but also 'remnant movement' structures like (10) where moved elements can contain moved elements (recursively, without bound).[8]

3. Merge: Conflated and persistent features

Minimalist grammars, as defined in the previous sections, partition the categories and the licensees into non-overlapping sets, but perhaps some properties of heads can enter into both selection and movement relations. For example, a verb might select a D, and a tense head might want that very same element to appear in its specifier because it is a D. The MGs above insist on having two features for these two different roles: category D can be selected, while only -D could be licensed by movement.

Inspired by Chomsky [20, §4.4.4] and Collins [27, §5.2], let's (i) conflate the categories and licensees, using only features f and selectors/licensors =f, and (ii) specify that some subset of these features is persistent (or 'interpretable'). (Non-persistent features are sometimes called 'formal'.) If features are ordered and merge steps do not erase persistent features, then everything following a persistent feature would be inaccessible. So we could remove the order, and assume that constituents have a set of features. That approach is common, but then when a head selects two constituents, what controls which constituent is selected first? To avoid that problem, let's keep ordered features but allow merge to optionally erase persistent features, so that when they have played all their roles in a derivation they can be deleted. For this approach, we leave the previous definition of em (6) unchanged; we modify the definition of im (8) so that it is triggered by =x,

(11) $$\mathbf{im}(t_1[\text{=x}]) \;=\; \begin{array}{c} > \\ t_2^M \quad t_1\{t_2[x]^M \mapsto \epsilon\} \end{array} \text{ if SMC,}$$

and for persistent features we add variants of (6) and (11) that apply only to persistent features, leaving them unerased. (Persistent features are underlined.)

(12) $$\mathbf{em'}(t_1[\text{=x}], t_2[\underline{\text{x}}]) = \begin{cases} \begin{array}{c} < \\ t_1 \quad t_2[\underline{\text{x}}] \end{array} & \text{if } |t_1| = 1 \\ \begin{array}{c} > \\ t_2[\underline{\text{x}}] \quad t_1 \end{array} & \text{otherwise,} \end{cases}$$

$$\mathbf{im'}(t_1[\text{=x}]) \;=\; \begin{array}{c} > \\ t_2[\underline{\text{x}}]^M \quad t_1\{t_2[\underline{\text{x}}]^M \mapsto \epsilon\} \end{array} \text{ if SMC}$$

Let's call these grammars 'conflated minimalist grammars' **(CMGs)**: merge is now defined by (6), (11), and (12).

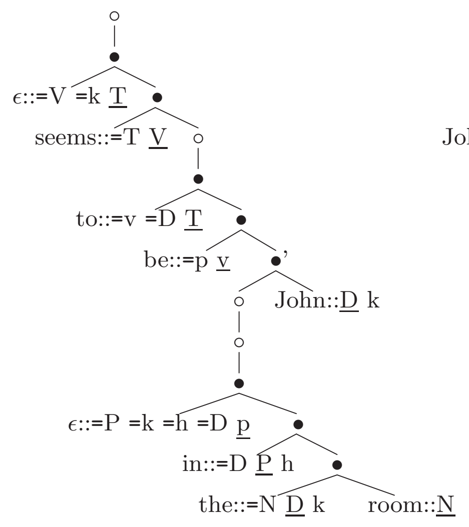
Example G2. In his discussion of persistent features, Collins [27, p.102] considers a structure that he gives schematically as follows:

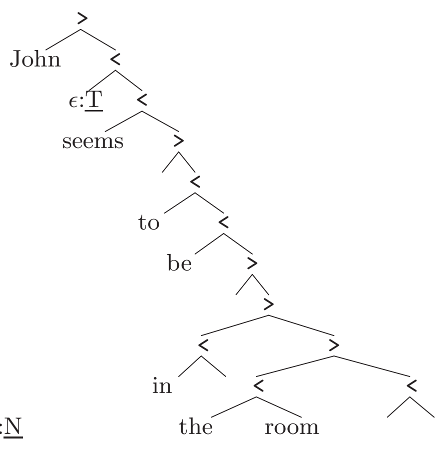
(13) $[_{\text{TP}}$ John$_i$ $[_{\text{T'}}$ T $[_{\text{VP}}$ seems $[_{\text{TP}}$ t_i $[_{\text{T'}}$ to be $[_{\text{SC}}$ t_i in the room$]]]]]]$,

The following tiny CMG lexicon suffices to derive structures of this form. (Persistent features are underlined, and remnant movement triggered by feature h is used get the preposition into the right place. Cf. §6.1 below on head movement).

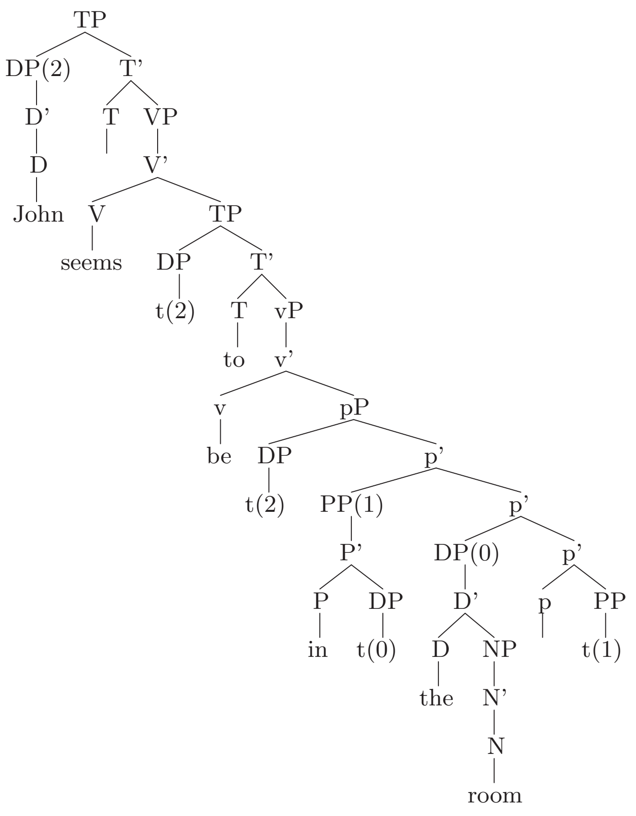
ϵ::=V =k $\underline{\text{T}}$	seems::=T $\underline{\text{V}}$	to::=v =D $\underline{\text{T}}$
be::=p $\underline{\text{v}}$	in::=D $\underline{\text{P}}$ h	ϵ:=P =k =h =D $\underline{\text{p}}$
the::=N $\underline{\text{D}}$ k	room::$\underline{\text{N}}$	John::$\underline{\text{D}}$ k.

The derivation shown below, left, yields the structure on the right:





The bare phrase structure reveals so little about the derivation, it can be useful to compute the corresponding redundant but more readable X-bar tree:



The MG finite partition property (9) still holds for CMGs: in useful CMG trees, there is a finite bound on the number of visible heads. And in fact, MGs and CMGs do not differ at all in the languages they define:

Theorem 2. *CMG≡MG.*

A proof is sketched in Appendix A.1, below. This result obviously means that the question of whether humans use CMGs rather MGs cannot be based simply on what expressions are in any particular language. In fact, it is not clear that an implementation of an CMG could be distinguished from an implementation of an MG: they might be different specifications of the very same implemented computation.[9]

Many other feature checking and feature percolation schemes are proposed in the literature (and we will consider one more in §4.2, below). Some are similar in expressive power and succinctness to the ones in MGs or CMGs; others are known to make the grammars as powerful as any computing device [50]. Obviously, the real challenge is to see through the notational variants to find the most restrictive characterizations that can provide insightful, explanatory models of human language abilities.

4. Merge: Locality

4.1. Phases

It is possible that certain categories make everything in their complements invisible to syntactic operations, acting rather like the 'bounding nodes' of earlier work in the GB tradition. We could call such categories 'phases', after the similar proposals in the literature [21,22,24,12,1]. For example, the categories v and C could be phases in this sense. The following simple condition, which we will call PIC after the similar formulations of 'phase impenetrability conditions' in the literature, obtains the desired effect in MGs as soon as the phase feature becomes visible, which will always happen exactly at the point when the projection of the phase, including all movements to specifier positions (the 'edge' of the category), has been completed:

(14) (PIC) Merge cannot apply to a tree $t[f]$ if f is a phase and if the complement of the head of $t[f]$ has any syntactic features.

The PIC restricts both em and im, so no outstanding conditions in the complement of any phase can ever appear in a successful derivation. Call MGs with this constraint 'phase-based minimalist grammars' (PMGs): they are (i) MGs as defined in §§1-2, (ii) extended with a specification of which categories (if any) are phases, and (iii) with PIC. We can also consider 'phase-based conflated minimalist grammars' (PCMGs) by imposing PIC on the CMGs of §3.

We can establish the following result, and easily adapt the standard MG parsing methods to the phase-based grammars:

Theorem 3. *PCMG≡PMG≡MG.*

A proof is sketched in Appendix A.2.

One of the intuitive ideas behind phase theory is that the material in the complements of phases is ready to be "sent to the interfaces" – so that no syntactic operations can affect the interpretation or the linear order of pronounced elements inside a phase complement. Notice that, at least with regard to the pronounced sequences, we can establish this as a theorem about PMGs (and PCMGs) too: since no outstanding syntactic features are allowed in phase complements, it follows immediately that nothing can affect the linear order of phrases there. Chomsky apparently has some additional point in mind [23, pp.16-17, for example]: perhaps phases could provide some analog of the finite partition property, so that, even without the assumption of SMC or anything like it, no search of unbounded structure would ever be needed to determine whether a derivational step can be taken (e.g. to find a matching head for im). For example, suppose that the amount of syntactically visible material in a phase (not counting the material inside contained phases) were finitely bounded; that could be significant for recognition or any other algorithm that required discovering derivations. But phases as defined above provide no such guarantee.[10]

4.2. Relativized minimality

The SMC given in (7) above blocks derivations in which there are two -wh elements competing for the same position; intuitively, allowing either to move would mean that the other would not be checked by the closest available +wh position. In a natural grammar, this principle could block constructions like this:[11]

(15) What$_i$ do you wonder how$_j$ [to solve t_i t_j]

Rizzi's 'relativized minimality' suggests a natural generalization of the SMC, extending intervention beyond matching features to broader classes [82, 81, 80]. Not only can one wh-phrase not move across another, but also, for example, in (16) one subject cannot move across another, and in (17) one adverb cannot move across another (examples 16-19 from [81]):

(16) * John$_i$ seems that it is likely t_i to win

(17) Rapidamente$_i$, i tecnici hanno (*probabilmente) risolto t_i
rapidly, the technicians have probably resolved
il problema (Italian)
the problem

However, it seems that an adverb can sometimes be extracted across an adverb, e.g., if it is being moved to a stressed focus position,

(18) RAPIDAMENTE$_i$, i tecnici hanno probabilmente risolto
RAPIDLY, the technicians have probably resolved
t_i il problema (non lentamente)
the problem (not slowly)

A similar kind of 'selective island' effect is also suggested by Obenauer's examples,

(19) a. [Combien de livres]$_i$ a-t-il beaucoup consultés t_i (French)
how-many of books has-he much consulted

b. *Combien$_i$ a-t-il beaucoup consultés [t_i de livres]

Notice also that in (16), (17), and (19b), the intervening elements are blocking the movements because of features that will have been checked and deleted in an MG or CMG derivation at the time when the offending movement wants to take place. So in order to determine this kind of intervention, several changes in the MG formalism are needed.

We adapt MGs to capture a number of Rizzi's insights as follows. *First*, we conflate selectors with probes, and licensees with categories, distinguishing some subset of these as persistent, as in CMGs. *Second*, in order to obtain intervention effects like those suggested by (16) and (17), since checked features seem to be relevant, the merge rule is modified so that, while the distinction between persistent and non-persistent features is maintained for checking relations, all features remain visible for intervention effects. Instead of marking progress on the requirements of lexical items by erasing features, we move a dot through the sequence. That is, lexical items will start with dot-initial feature list $\bullet\alpha_1\alpha_2\ldots\alpha_n$; when α_1 is checked and becomes invisible to any other probe, we get the sequence $\alpha_1\bullet\alpha_2\ldots\alpha_n$; and when α_1 is checked and persists (e.g. for the kind of cyclic movements mentioned in §3), the feature sequence is left unchanged. The dot is similarly moved across later features, marking the unique feature that is visible to a probe/selector while leaving all the features available for the determination of blocking effects. In this system, the notation $t[f]$ refers to a tree whose head has feature f immediately following the dot. *Third*, for any tree t, let *type* be a function mapping each basic feature f to features that will block movement of $t[f]$. And *finally*, potential interveners are defined in terms of c-command as follows.[12] For any subtree t_2 of tree t_1, let $\mathit{cover}(t_2)$ be the set of features of heads t_3 such that t_3^M c-commands t_2^M. With these definitions, we formulate this replacement for the SMC:

(20) (RMC) Im applies to $t_1[\texttt{=}f]$ only if (i) $t_1[\texttt{=}f]$ has exactly one subtree $t_2[f]$, and (ii) $\mathit{cover}(t_2[f]) \cap \mathit{type}(f) = \emptyset$.

Let's call grammars that are constrained in this way "relativized minimalist grammars" (RMGs). In this kind of RMG it is possible to require that a =f movement of an adverb is blocked by another intervening adverb, while a different kind of movement triggered by a different feature =g can move the same adverb without being blocked. RMGs have the same expressive power as MGs, and standard parsing methods extend to them (Appendix A.3):

Theorem 4. *RMG≡MG.*

In recent work, Rizzi has proposed another, rather different restriction on movement. The reader will have noticed that in example G2, above, an 'abstract case' feature -k was checked in the tensed matrix clause. Rizzi observes that the subject requirement is not always associated with the case system in this way, and proposes a more general, semantically-motivated account of why clauses need subjects and why subjects of tensed clauses tend not to be movable [83,84]. Adapting his account slightly to the terms of the present framework:

(21) Certain designated positions are 'criterial' in the sense that they are dedicated to a particular interpretive property. These positions may be identified as the specifiers created by +f features for f = q, top, foc,. . .

(22) (Criterial freezing) Nothing can be moved from a criterial position.

Rizzi shows how a range of subject-object asymmetries follow from the fact that there is a 'subject criterion' but not an 'object criterion'.[13] In the present framework, criterial freezing is achieved simply by making the criterial features non-persistent, and requiring (perhaps for semantic reasons) that these features always appear as the last feature of any head.[14]

4.3. Multiple movements and multiple agree

The theory of merge and its locality restrictions needs to allow for wh-in-situ and multiple wh-fronting [88,61,13,11]. Wh-in-situ can be allowed in simply by assuming that they do not have formal features which must be checked by movement. Multiple wh-movement poses trickier problems for SMC (and also RMC and related constraints): they do not allow im to apply when there are two -wh features. There are several strategies that might be worth pursuing. First, an SMC-like constraint might be tenable if we refine the wh-features (e.g. wh-nom, wh-acc, wh-manner,. . .) so that each wh-movement is triggered by a different wh-feature. Another strategy introduces a special 'clustering' or 'absorption' case of merge which, upon encountering a second -wh phrase in a derivation, immediately merges those two phrases into a single one, leaving a single -wh feature [34,32,74]. This idea might fit with the special ordering ('tucking in') found in some multiple-wh constructions, noted for example in [78]. A third strategy relaxes SMC-like constraints to allow a more expressive formalism (perhaps similar to the '-SpIC -SMC' formalism mentioned [31]). It is not yet clear which approach can best fit the facts.

As noted in §6.2 below, it is easy to extend MGs to allow 'feature movement', that is, instances of im that check features in the usual way but which do not move anything. Operations of this sort have been proposed in analyses of agreement relations. But in that case, multiple agreement relations will pose the same puzzle as multiple movements.

5. Merge: Linearization

Merge is defined in §2 so that complements are attached to the right of the selecting head, while specifiers are attached on the left. This underlying 'SVO' order is stipulated in the definition of our merge operation, but is conceivably the result of some other forces on linearization. We will consider that possibility but first, it is worth observing the unsurprising fact that all orders can obtained by movement. It is easy to establish that the MG definable languages are 'closed with respect to reversal'. That is, for any minimalist grammar G defining language L(G), the result of reversing every sentence of that language is a language L(G'), where G' is another minimalist grammar. Is this a signal that the formalism is overly general? Perhaps not.

Greenberg's Universal 20 observes that certain orders of the basic elements of determiner phrases are common across languages, while others are extremely rare or unattested [37]. In a recent assessment of Greenberg's proposal, Cinque [25] reports that only 14 of the possible 24 orderings of [Dem Num Adj N] are attested. These ordering facts are sometimes offered as evidence for the assumptions like the ones made in the definition of merge in §2, but if all orders can be derived by movement, it is natural to wonder whether all hope of explaining Universal 20 and related facts is lost. Perhaps not.

While MGs are closed with respect to reversal, nevertheless, some orders are easier to define than others. That is, some orders of elements require more steps in their derivation. In particular, suppose that head 1 selects head 2, head 2 selects head 3, and head 3 selects head 4. Then we can derive the order 1234 using em only. Adding pairs of features (+f,-f) triggering movement to these heads, it turns out that we can derive many other orders, but not all of them. For example, we cannot derive the order 3214, as can be seen by trying all the possibilities. (Trying all the possibilities is quite tedious by hand, but is much easier when the process is automated.) Notice first that movement never changes the root category, so given our assumptions about selection, 1 must be the root of all derivable orders. If we just let head 1 trigger movement of head 3 to its specifier, we get the order 2341. Since the 4 is in the wrong place, we can try to move it first: 4 can move to the specifier of heads 1, 2, or 3, but in none of these positions can we separate it from 2 and 3. (Appendix A.4 shows MG grammars for derivable orders.)

Given the assumption that the selection order is 1234, and assuming no other heads are introduced, only 16 out of the 24 orders can be obtained by movement, and 14 of those 16 are the ones that Cinque reports as attested. It is hard to believe that this is an accident! So one might be tempted to assume that these linear asymmetries are coming from the underlying SVO order. But that assumption would be a mistake, as recently pointed out by Abels and Neeleman [3,2]. They point out that the patterns observed by Cinque are well accounted for even when there can be heads and specifiers on either side of the head.

One way to see this point is to consider another variant on MGs in which im is unchanged, but the feature =X always triggers attachment on the right and X= triggers attachment on the left. That is:[15]

(23) $$\mathbf{em}(t_1[\alpha], t_2[\mathtt{x}]) = \begin{cases} \overset{\texttt{<}}{\diagup\!\diagdown}\\[-2pt] t_1 \quad t_2 & \text{if } \alpha \text{ is } \mathtt{=x} \\ \overset{\texttt{>}}{\diagup\!\diagdown}\\[-2pt] t_2 \quad t_1 & \text{if } \alpha \text{ is } \mathtt{x=} \end{cases}$$

Let's call grammars defined by by (5), (8) and (23) 'directional MGs' (DMGs). Using the same proof strategy used for Theorems 2 and 3, it is easy to establish

Theorem 5. *DMG≡MG,*

but the derivational complexities of various surface orders obtainable in DMGs and MGs can differ. Keeping the previous assumption that we have exactly four heads with the selection order 1234, DMGs derive the 8 of the 24 possible orders using em only, and only 8 of the other orders are derivable. The following table compares the minimal derivational complexities of the various orders in MGs and DMGs with Cinque's assessment of the typological frequencies of each. To facilitate comparison, for Cinque's classification, we write 0 for 'unattested' orders, 1 for orders found in 'very few' languages, 2 for 'few', 3 for 'many', and 4 for 'very many'. For the grammars, since all derivable orders are obtained with 3 or fewer licensees, we use 0 to indicate underivable orders and otherwise use the quantity $(4 - \ell)$ where ℓ is the minimum number of licensees needed to get the order. (See grammars in Appendix A.4.)

order	Cinque	MG	DMG	order	Cinque	MG	DMG
1234	**4**	**4**	**4**	1324	**0**	**0**	**0**
1243	**3**	**3**	**4**	1342	**1**	**3**	**4**
1423	**1**	**3**	**3**	1432	**3**	**2**	**4**
4123	**2**	**3**	**3**	4132	**1**	**2**	**3**
2134	**0**	**0**	**0**	2314	**0**	**0**	**0**
2143	**0**	**0**	**0**	2341	**1**	**3**	**4**
2413	**0**	**0**	**0**	2431	**2**	**2**	**4**
4213	**0**	**0**	**0**	4231	**2**	**2**	**4**
3124	**0**	**0**	**0**	3214	**0**	**0**	**0**
3142	**0**	**2**	**2**	3241	**0**	**1**	**2**
3412	**1**	**3**	**3**	3421	**1**	**2**	**4**
4312	**2**	**2**	**3**	4321	**4**	**1**	**4**

First, comparing the underivable and unattested orders, we see that MGs and DMGs are identically in good agreement with Cinque's results. And in addition, there is a tendency for rarer orders to be harder to derive. Given the nature of Cinque's typological classification and the way we have coded derivational complexity here, it is important to avoid reading too much into the quantitative details, but calculating Pearson correlation coefficients, we find a correlation of 0.62 between the MG ranks and Cinque's, and a correlation of 0.75 between the DMG ranks and Cinque's. So we see that DMGs fit the typological data without the assumption of a rigid SVO order.

Up to this point, we have been coding linear order in the derived trees, but this is slightly redundant. Notice for example that in the simple MGs, both em and im put specifiers on the left. There is a missed generalization there, and we may find other simplifications of the grammar if we separate the determination of linear order from the calculation of hierarchical structure. A simple and elegant idea about this is explored by Kayne [48] and has been much discussed. From our computational perspective, consider the grammars that result from retracting our assumption that the derived trees created by em in (6) and im in (8) are ordered. Then immediately, the two cases given in (6) collapse into one. We can consider what function properly maps the unordered, derived trees into linearly ordered structures. The theory of tree transducers [52, 57] provides a natural framework in which the computational properties of such functions can be studied.

6. More variations

6.1. Head movement

The nature of head movement remains controversial, but many traditional instances of head movement can be captured with a simple extensions of minimalist grammars (MGH), studied in [91–93,65]. A certain MG equivalent version of a 'mirror' theory inspired by Brody's proposals has also been carefully studied in [49]. These extensions too are all weakly equivalent to MGs. Intuitively, the effects of head movement, and of mirror theory, can be achieved by phrasal remnant movement. Many mysteries remain.

6.2. LF and PF movements

Many minimalist proposals consider the possibility that all constraints on syntax should come from the LF and PF interfaces, with the action of merge between these two interfaces kept as simple as possible This leads some to ask: if im is acting to check a feature, why does this ever involve movement of structure with phonetic material in it? This possibility was already familiar in analyses of 'covert' movement at least since [17]. It is easy to add feature checking without overt movement of structure to MGs, a kind of covert 'LF movement', to MGs, as studied in the early work [91,62]. If some features are interpretable while others are not, as briefly mentioned in §3, above, then there could also be processes that leave the interpretable features behind while moving phonetic features, 'PF movement.' Perhaps certain head movement phenomena and certain instances of scrambling should be treated as PF movements.

6.3. Adjunct merge

The external merge rule introduced in §2 is sometimes called 'argument merge,' and there are various proposals for another kind of operation for merging adjuncts [56,19,29,30]. These are usually proposed in order to allow a more natural account of the certain 'reconstruction' effects found with arguments but not with adjuncts. A proposal of this kind has been formally modeled in [33].

6.4. Sideward movement

The MG variants defined above all allow merge to select freely from the lexicon. Some proposals compare derivations that involve exactly the same multiset of lexical elements.[16] Phases provide a different kind of domain, sometimes regarded as a "subarray" of constituents [21,22,24], from which the structure building functions can select their arguments. On some conceptions, these intermediate workspaces admit a kind of 'sideward movement' [99,73,45,10,26] which moves an element from one tree to another which has not yet been merged or adjoined. Increasing the domain of merge in this way seems to open the way to more unified accounts of various phenomena. Some preliminary studies of this kind of domain have been undertaken in [96,95], again MG equivalent systems.

6.5. Copy and delete

The MG variants considered so far have left nothing behind when something moves. In the implementation of standard parsing methods, this means simply that syntactic relations can be constructed from non-adjacent parts. There is no particular mystery in that; in all perceptual domains we sometimes recognize a unity among components that are (spatially or temporally) non-adjacent. And we have various kinds of discontinuity in artificial languages that we design for our own convenience; mutually dependent type declarations in many programming languages need not be adjacent, and are usually rejected or flagged if they are vacuous [58]. But familiar arguments suggest that moved phrases sometimes seem to be interpreted as if they are their original positions [17,60,19], and sometimes we even seem to see all or part of the phonetic contents of the moved phrase in the original positions too [51,14].

Suppose we keep the definition of em unchanged from §2, but identify a subset of features that trigger the use of this special additional case of im, defined in terms of a function g from trees to trees:

(24) $$\mathbf{im}(t_1[\texttt{+x}]) \;=\; \begin{tikzcd}[column sep=small, row sep=small] & > \arrow[dl, dash] \arrow[dr, dash] & \\ t_2^M & & t_1\{t_2[\texttt{-x}]^M \mapsto g(t_2[\texttt{-x}]^M)\} \end{tikzcd} \qquad \text{if SMC}$$

Now we obtain different kinds of grammar depending on the function g. If g maps every tree to the empty tree ϵ, this is exactly the original im. But now consider, for example, the function that g leaves the structure and phonetic contents of each tree untouched, removing only any outstanding syntactic features. Let's call these grammars 'minimalist grammars with copying' (MGC). MGCs have quite different computational properties from the previous MG variants, as discussed in [51].[17] Again they define languages in an efficiently recognizable class that has been studied: so-called 'parallel multiple context free grammars' (PMCFGs) [90]:

Theorem 6. $CF \subset \boxed{MCFG \equiv MG} \subset \boxed{MGC \subseteq PMCFG} \subset CS.$

7. Next steps

The formal, computational studies of apparently diverse ideas in the minimalist program reveal some surprising properties and especially commonalities. (Cf. [31], a survey emphasizing some significant *differences* among minimalist proposals.) There will always be a big gap between new empirical speculations and what is well understood, but we can seek results that apply to large families of related proposals whenever possible, identifying common themes if only approximately. Further study will certainly illuminate important distinctions that have been missed here and bring deeper understanding of the fundamental mechanisms of human language.

Acknowledgements: Thanks to Chris Collins, Greg Kobele, and Jens Michaelis for help with this material.

Notes

[1]When Chomsky [23, p.23] stipulates that the probe/selector α remains the label, as in (1), he suggests that this is the 'simplest' idea, since then no other label needs to be sought. What does that mean? Notice that the label is presumably represented somehow, so the size of that representation will contribute to memory demands, and the nature of that representation will be relevant for ensuing computational steps. It is difficult to evaluate such matters in a principled and relevant way, since the time and space efficiency of an algorithm (that is, how many steps are taken on each input, and how much memory is used) is determined relative to a computational architecture (what counts as a step; how is the memory accessed; can steps be taken in parallel; etc.) – see for example, [102, 75]. Perusing any standard text on algorithms like [28], one finds many surprises, and very many results that naive, informal intuition could not have led us to, so casual claims about complexity should be considered with appropriate care.

[2]Chomsky [20, p.245] says that the set in (1) is simpler than the tree in (2), but it is not clear what the relevant notion of simplicity is. Perhaps the set is assumed simpler since it involves just 4 objects – α, β, $\{\alpha, \beta\}$, and $\{\alpha, \{\alpha, \beta\}\}$ – with a membership relation as defined by some standard set theory. Considering the tree on the other hand, we have 3 nodes, each labeled, and a dominance relation. So at this point, we will regard the tree and the set as notational variants, until we have found a reason to regard some distinguishing property of the notations as linguistically relevant.

[3]It is possible to assume that whatever requirements are imposed on the 'selection' relation are not part of the definition of merge, but result from the action of other constraints. Obviously, this could provide a notational variant of the formalism given here, or could be quite different, depending on the nature of the constraints.

[4]Along with TAG and CCG languages, the MG definable languages are 'mildly context sensitive' in the sense defined by Joshi [47]. This is discussed in [62] and [94].

[5]As discussed in [89, §IV], [63, §5.2], [59, §3], a class of languages is a 'substitution-closed full abstract family of languages' just in case it is closed with respect to finite unions, finite products, Kleene star, arbitrary homomorphisms, inverse homomorphisms, and substitutions.

[6]The recognition problem for MCFGs or MGs can be solved 'efficiently' in the sense that the number of steps required can be bounded by a polynomial function of a natural measure of the input size, assuming that the steps are taken serially on a 'random access machine' of the kind mentioned in [102]. Harkema presents a CKY-like algorithm for MG recognition that is guaranteed to need no more than $\mathcal{O}(n^{4m+4})$ steps [40,42], where n is the length of the input and m is a constant depending on the grammar (specifically, on the number of licensees that can occur in a derivation). Standard parsing algorithms like CKY and Earley's present many opportunities for parallelism, and so it is no surprise that much faster execution can be obtained when enough parallelism is available [54,100,43].

[7]Some of Ristad's [79] arguments can be understood as showing that languages defined by GB syntax are more complex than context free grammars, but these

arguments include the indexing mechanisms of binding theory. Cf. the discussion of various sources of complexity in [4].

[8]Although simplistic ‘slash-passing’ grammars are not easily extended to the MG class (or to the MGC class mentioned in §6.5 below), the ‘attribute grammar’ formalism of [8] might be regarded as similar, and it can easily be extended to MGs, as shown by [67].

[9]Familiar programs are implemented by ‘compiling’ them into codes that can be ‘directly’ executed by a computer, and it is reasonable to expect the neural implementation of our high level linguistic performance models to be at least as involved. But the question of what should count as an implementation of a given program (particularly with ‘optimizing’ compilers) is not clear; addressing this question in the case of linguistic computations will certainly comprise part of the scientific problem. See for example [7,68].

[10]Chesi [15] notes that although phases have been argued to be of unbounded size (p.48n32), he can guarantee they are finite by stipulating a fixed finite bound on the number of arguments allowed in each phase and by assuming that the functional categories in each phase are not recursive. If these assumptions are warranted, they could of course be adopted here too, potentially giving us an alternative route to the finite partition property or something similar; the challenge is to defend those assumptions.

[11]We leave aside the argument-adjunct asymmetries in such movements, but consider them in [97], and we postpone the discussion of multiple wh-extractions to §4.3 below.

[12]As usual, the dominance relation is reflexive (so in any tree, every node dominates itself), and in any tree with subtrees t_1, t_2, t_3, subtree t_1 *c-commands* subtree t_2 if and only if the root of t_2 is dominated by a sister of the root of t_1.

[13]Rizzi and Shlonsky observe that sometimes the subject *can* move, noting an example in Imbabura Quechua and other things, but also the simple English *Who came?* In the English question, they suggest, the ϕ features of the finite clause and valued by *who* suffice to satisfy the subject criterion, so that *who* can move to the C domain: “So, Fin+Phi offers a kind of bypassing device,. . . allowing the thematic subject endowed with the wh- (or some other A’-) feature to move higher.”

[14]Abels observes that a ban on improper movement can also be enforced with restrictions on feature ordering [2].

[15]The DMGs introduced here generate more orders than allowed by the assumptions of Abels and Neeleman [3,2]: Abels and Neeleman disallow movements that do not affect the noun, for reasons I do not understand. I think it is more interesting to compare all movements that involve only elements of the DP structure, as I do here, but the conclusions I mention here are the same in either case.

[16]Multisets are often called ‘numerations’ by linguists; computer scientists sometimes call them ‘heaps’ or ‘bags.’ Multisets of elements of a set Lex can be formalized as functions from Lex into the natural numbers; each lexical item is mapped to the number of times it occurs.

[17]As discussed by [51] and [94], these languages are *not* ‘mildly context sensitive’ in the sense defined by Joshi [47]. Cf. note 4.

Appendix: MG variants and MCFGs

A.1. CMG≡MG

To prove this claim from page 11, we show (1) that CMG⊆MG and (2) MG⊆CMG.

To establish (1), it suffices to show establish (3) CMG⊆MCFG since it is already known that (4) MCFG⊆MG [64,41]. In fact, the proof of (3) can be a very minor variation of the proof of MG⊆MCFG given by Michaelis, so we sketch the main ideas here and refer the reader to Michaelis's earlier proof [62,63] for full details.

(3) can be established in two steps. First we show (3a) every CMG using merge is exactly equivalent to a grammar defined over tuples of categorized strings, and then (3b) we provide a simple recipe for converting the grammars over tuples of categorized strings into an exactly equivalent MCFG over tuples of strings. For (3a), a minor variation of the presentation in [98] suffices. The conversions between grammar formalisms needed for (3a-b) are very simple and easily automated [38].

For (3a), we define,

(25) vocabulary $\Sigma = \{$ every,some,student,...$\}$
types $T = \{$`::`, `:`$\}$ ('lexical' and 'derived', respectively)
syntactic features F of two kinds:

C, T, D, N, V,. . . , wh, case,. . .	(basic features)
=C, =T, =D, =N, =V,. . . , =wh, =case,. . .	(selectors/probes)

persistent features $= P \subseteq F$
chains $C = \Sigma^* \times T \times F^*$
expressions $E = C^*$
lexicon $Lex \subset \Sigma^* \times \{::\} \times F^*$, a finite set.

Then we define *merge* as the union of the following 7 functions, (3 for em, 2 for im, and then for persistent features: em3' and im2').

(26) For $s, t \in \Sigma^*$, for $\cdot \in \{:, ::\}$, $\gamma, \gamma' \in F^*$, $\delta \in F^+$, and where $\alpha_1, \ldots, \alpha_k, \iota_1, \ldots, \iota_l$ $(0 \leq k, l)$ are any chains, define:

$$\frac{s :: \texttt{=}f\gamma \qquad t \cdot f, \alpha_1, \ldots, \alpha_k}{st : \gamma, \alpha_1, \ldots, \alpha_k} \text{ em1: lexical item selects a non-mover}$$

$$\frac{s : \texttt{=}f\gamma, \alpha_1, \ldots, \alpha_l \qquad t \cdot f, \iota_1, \ldots, \iota_k}{ts : \gamma, \alpha_1, \ldots, \alpha_l, \iota_1, \ldots, \iota_k} \text{ em2: derived item selects a non-mover}$$

$$\frac{s \cdot \texttt{=}f\gamma, \alpha_1, \ldots, \alpha_l \qquad t \cdot f\delta, \iota_1, \ldots, \iota_k}{s : \gamma, \alpha_1, \ldots, \alpha_l, t : \delta, \iota_1, \ldots, \iota_k} \text{ em3: any item selects a mover}$$

$$\frac{s : \texttt{=}f\gamma, \alpha_1, \ldots, \alpha_{i-1}, t : f, \alpha_{i+1}, \ldots, \alpha_k}{ts : \gamma, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k} \text{ im1: final move of licensee}$$

$$\frac{s : \texttt{=}f\gamma, \alpha_1, \ldots, \alpha_{i-1}, t : f\delta, \alpha_{i+1}, \ldots, \alpha_k}{s : \gamma, \alpha_1, \ldots, \alpha_{i-1}, t : \delta, \alpha_{i+1}, \ldots, \alpha_k} \text{im2: nonfinal move of licensee}$$

where

(tuple-SMC) none of the chains $\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k$ in im1, im2 or im2' has f as its first feature.

These first 5 functions apply regardless of whether the features involved are persistent. The functions for only persistent features (indicated by underlining) are similar, except the feature is not deleted. Since the undeleted feature is then available for internal merge, we have only the 'mover' cases:

$$\frac{s \cdot \texttt{=}f\gamma, \alpha_1, \ldots, \alpha_l \qquad t \cdot \underline{f}\gamma', \iota_1, \ldots, \iota_k}{s \cdot \gamma, \alpha_1, \ldots, \alpha_l, t : \underline{f}\gamma', \iota_1, \ldots, \iota_k} \text{ em3': any item selects a mover}$$

$$\frac{s : \texttt{=}f\gamma, \alpha_1, \ldots, \alpha_{i-1}, t : \underline{f}\gamma', \alpha_{i+1}, \ldots, \alpha_k}{s : \gamma, \alpha_1, \ldots, \alpha_{i-1}, t : \underline{f}\gamma', \alpha_{i+1}, \ldots, \alpha_k} \text{im2': nonfinal move of licensee}$$

Define the structures S(G)=closure(Lex,{em1,em2,em3,im1,im2,em3',im2'}). The completed structures are those expressions $w \cdot C$, where C is the 'start' category and $\cdot \in \{:, ::\}$. The sentences L(G) $= \{w| \ w \cdot C \in S(G)$ for some $\cdot \in \{:, ::\}\}$. Now it is easy to show by an induction on derivation depth that derivations from these grammars are isomorphic to the ones over trees, with the same lexical items and the same yields. For (3b) we convert these grammars to exactly equivalent MCFGs as in Michaelis's earlier proof: the tuples of categorized strings can be represented as tuple of strings with a single simple category, where the number of simple categories needed is finite by (9).

To complete the proof it remains only to show (2) MG⊆CMG. Given any MG, we remove any useless lexical items, and then produce a CMG simply by changing every +f in the MG to =f, and changing every -f to f (after renaming if necessary so that none of these new =f and f features are the same as ones already in the grammar). We let the set of persistent features $P = \emptyset$. These grammars will produce isomorphic derivations. Although the features are conflated in the CMG, since none of them are persistent, every step will delete a pair of features, and so every em and im step in the CMG will correspond to the same kind of step in the MG.

A.2. PCMG≡PMG≡MG

This claim from page 12 can be proven with the same kind of strategy used above in A.1 and in [62]. Here we consider only PCMG≡MG, since PMG≡MG can be established in an exactly analogous fashion. We show (1) that PCMG⊆MG and (2) MG⊆PCMG.

As in A.1, to establish (1), it suffices to show (3) PCMG⊆MCFG, which can be done in two steps: showing (3a) every PCMG using merge is exactly equivalent to a grammar defined over tuples of categorized strings, and then showing (3b) a simple recipe for converting the grammars over tuples of categorized strings into an exactly equivalent MCFG over tuples of strings.

Adapting step (3a) from A.1 for CMGs with phases, we adopt the definitions in (25), adding only a specification of a set of phases $Ph \subseteq F$. Merge is unchanged except that we impose the following condition on em (that is, on em1, em2, em3, and em3'):

(tuple-PIC) em cannot apply if $f \in Ph$ and $k > 0$.

It is easy to show that this grammar on tuples of categorized strings is equivalent to the corresponding PCMG, and then (3b) can be established as in A.1 and in [62]. The other direction, (2), is trivial, since a CMG is simply a PCMG where $Ph = \emptyset$.

A.3. RMG≡MG

The results in the previous appendices A.1 and A.2 involve minor adjustments in the basic idea from Michaelis's [62], but this result requires a more substantial innovation. A complete presentation of the proof is given in [97], where the extension of RMGs to head movement as an instance of internal merge [81,85] is also discussed, together with RMG parsing methods. Here we just sketch the main idea of the proof.

The proof is split in the usual way: (1) RMG⊆MG and (2) MG⊆RMG, where (1) is the challenging step. To establish (1), it suffices to show (3) RMG⊆MCFG, which can be done in two steps: showing (3a) every RMG is exactly equivalent to some intermediate grammar G, and then showing (3b) a simple recipe for converting each such intermediate G into an exactly equivalent MCFG over tuples of strings. For previous results, the intermediate grammar G was a grammar over tuples of categorized strings (*of bounded size*), but for RMGs, a straightforward extension of that idea will not suffice. In RMGs, we need to keep track of both the active sequences of features and potential interveners. Notice that the interveners for each category can be represented as a subset of the finite set of features, and so this can be added to our categories without threatening the finite partition property. The problem is that when a remnant moves, the set of potential interveners for both the remnant and every moving part of the remnant changes, because the c-commanders of those elements will change. So the most natural proof strategy is to let the intermediate grammars be defined over trees, as RMGs are, but use trees for which it is evident that the finite partition property holds. This can be done because we can let the leaves of the tree be exactly the categorized strings that would have been used in a tuple-based grammar – and we know the length of these tuples is bounded – and then use a binary tree structure over these leaves to indicate hierarchical relations and potential interveners. So then in the MCFG the categories are not tuples of feature sequences but trees with k leaves, and since k is bounded and the number of binary trees and intervener specifications is also bounded, the number of categories in the MCFG is finitely bounded too, as required.

A.4. Derivable permutations of 1234 in MGs and DMGs

In the following grammars, we let the 'phonetic forms' of the heads 1, 2, 3, 4 be the same as their categories. Many orders can be obtained in multiple equally simple ways, only one of which is shown. We write 'nd' for 'not derivable'. (Obviously, all orders are derivable if we allow additional heads, as pointed out in the text.)

order	MG				DMG			
1234	1::=2 1	2::=3 2	3::=4 3	4::4	1::=2 1	2::=3 2	3::=4 3	4::4
1243	1::=2 1	2::=3 2	3::=4 +4 3	4::4 -4	1::=2 1	2::=3 2	3::4= 3	4::4
1423	1::=2 1	2::=3 +4 2	3::=4 3	4::4 -4	1::2= +4 1	2::=3 2	3::4= 3	4::4 -4
4123	1::=2 +4 1	2::=3 2	3::=4 3	4::4 -4	1::=2 +4 1	2::=3 2	3::=4 3	4::4 -4
1324	nd				nd			
1342	1::=2 1	2::=3 +3 2	3::=4 3 -3	4::4	1::=2 1	2::3= 2	3::=4 3	4::4
1432	1::=2 1	2::=3 +3 2	3::=4 +4 3 -3	4::4 -4	1::=2 1	2::3= 2	3::4= 3	4::4
4132	1::=2 +4 1	2::=3 +3 2	3::=4 3 -3	4::4 -4	1::=2 +4 1	2::3= 2	3::=4 3	4::4 -4
2134	nd				nd			
2143	nd				nd			
2413	nd				nd			
4213	nd				nd			
2314	nd				nd			
2341	1::=2 1	2::=3 +3 2	3::=4 3 -3	4::4	1::2= 1	2::=3 2	3::4= 3	4::4
2431	1::=2 +2 1	2::=3 2 -2	3::=4 +4 3	4::4 -4	1::2= 1	2::=3 2	3::4= 3	4::4
4231	1::=2 +2 1	2::=3 +3 2 -2	3::=4 3	4::4 -4	1::2= +4 1	2::=3 2	3::4= 3	4::4 -4
3124	nd				nd			
3142	1::=2 +3 1	2::=3 +4 2	3::=4 3 -3	4::4 -4	1::=2 +3 1	2::=3 +4 2	3::=4 3 -3	4::4 -4
3412	1::=2 +3 1	2::=3 2	3::=4 3 -3	4::4	1::=2 +3 1	2::=3 2	3::=4 3 -3	4::4
4312	1::=2 +3 1	2::=3 2	3::=4 +4 3 -3	4::4 -4	1::=2 +3 1	2::=3 2	3::4= 3 -3	4::4
3214	nd				nd			
3241	1::=2 +4 +2 1	2::=3 +3 2 -2	3::=4 3 -3	4::4 -4	1::=2 +4 +2 1	2::3= 2 -2	3::=4 3	4::4 -4
3421	1::=2 +2 1	2::=3 +3 2 -2	3::=4 3 -3	4::4	1::2= 1	2::3= 2	3::=4 3	4::4
4321	1::=2 +2 1	2::=3 +3 2 -2	3::=4 +4 3 -3	4::4 -4	1::2= 1	2:: 3= 2	3::4= 3	4::4

References

1. Abels, K. *Successive Cyclicity, Anti-Locality, and Adposition Stranding.* PhD thesis, University of Connecticut, 2003. http://ling.auf.net/lingBuzz/000049.
2. Abels, K. Towards a restrictive theory of (remnant) movement: Improper movement, remnant movement, and a linear asymmetry. *Linguistic Variation Yearbook 2007 7* (2007), 53–120.
3. Abels, K., and Neeleman, A. Universal 20 without the LCA. Ms., University College, London, 2006.
4. Barton, G. E., Berwick, R. C., and Ristad, E. S. *Computational Complexity and Natural Language.* MIT Press, Cambridge, Massachusetts, 1987.
5. Berwick, R. C. Computational complexity of lexical functional grammar. In *Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics, ACL'81* (1981), pp. 7–12.
6. Berwick, R. C., and Weinberg, A. S. *The Grammatical Basis of Linguistic Performance: Language Use and Acquisition.* MIT Press, Cambridge, Massachusetts, 1984.
7. Blass, A., Dershowitz, N., and Gurevich, Y. When are two algorithms the same? University of Michigan, Tel Aviv University, and Microsoft Research. http://arxiv.org/abs/0811.0811, 2008.
8. Bloem, R., and Engelfriet, J. A comparison of tree transductions defined by monadic second order logic and by attribute grammars. *Journal of Computer and System Sciences 61*, 1 (2000), 1–50.
9. Boeckx, C. Agree or attract? a relativized minimality solution to a proper binding condition puzzle. In *Theoretical Approaches to Universals*, A. Alexiadou, Ed. John Benjamins, Philadelphia, 2002, pp. 41–64.
10. Boeckx, C., and Hornstein, N. Movement under control. *Linguistic Inquiry 35*, 3 (2004), 431–452.
11. Bošković, Ž. On multiple wh-fronting. *Linguistic Inquiry 33*, 3 (2002), 351–383.
12. Bošković, Ž. Agree, phases, and intervention effects. *Linguistic Analysis 33* (2003), 54–96.
13. Bošković, Ž. On multiple feature-checking: multiple wh-fronting and multiple head-movement. *forthcoming* (2007).
14. Bošković, Ž., and Nunes, J. The copy theory of movement. In *The Copy Theory of Movement*, N. Corver and J. Nunes, Eds. John Benjamins, Philadelphia, 2007.
15. Chesi, C. An introduction to phase-based minimalist grammars: Why *move* is top-down from left-to-right. Tech. rep., Centro Interdepartmentale di Studi Cognitivi sul Linguaggio, 2007.
16. Chomsky, N. *Aspects of the Theory of Syntax.* MIT Press, Cambridge, Massachusetts, 1965.
17. Chomsky, N. Conditions on rules of grammar. *Linguistic Analysis 2* (1976), 303–351.
18. Chomsky, N. *Lectures on Government and Binding.* Foris, Dordrecht, 1981.
19. Chomsky, N. A minimalist program for linguistic theory. In *The View from Building 20*, K. Hale and S. J. Keyser, Eds. MIT Press, Cambridge, Massachusetts, 1993.
20. Chomsky, N. *The Minimalist Program.* MIT Press, Cambridge, Massachusetts, 1995.
21. Chomsky, N. Minimalist inquiries: The framework. In *Step by Step: Essays on Minimalism in Honor of Howard Lasnik*, R. Martin, D. Michaels, and J. Uriagereka, Eds. MIT Press, Cambridge, Massachusetts, 2000, pp. 89–155.

22. Chomsky, N. Derivation by phase. In *Ken Hale: A Life in Language*, M. Kenstowicz, Ed. MIT Press, Cambridge, Massachusetts, 2001.
23. Chomsky, N. Approaching UG from below. In *Interfaces + Recursion = Language? Chomsky's Minimalism and the View from Syntax-Semantics*, U. Sauerland and H.-M. Gärtner, Eds. Mouton de Gruyter, NY, 2007, pp. 1–30.
24. Chomsky, N. On phases. In *Foundational Issues in Linguistic Theory: Essays in Honor of Jean-Roger Vergnaud*, R. Freidin, C. P. Otero, and M. L. Zubizarreta, Eds. MIT Press, Cambridge, Massachusetts, 2008.
25. Cinque, G. Deriving Greenberg's Universal 20 and its exceptions. *Linguistic Inquiry 36*, 3 (2005), 315–332.
26. Citko, B. On the nature of merge: external merge, internal merge, and parallel merge. *Linguistic Inquiry 36* (2005), 475–496.
27. Collins, C. *Local Economy*. MIT Press, Cambridge, Massachusetts, 1997.
28. Cormen, T. H., Leiserson, C. E., and Rivest, R. L. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, 1991.
29. Epstein, S. D., Groat, E. M., Kawashima, R., and Kitahara, H. *A derivational approach to syntactic relations*. Oxford University Press, NY, 1998.
30. Fox, D. On logical form. In *Minimalist Syntax*, R. Hendrick, Ed. Blackwell, Oxford, 2003, pp. 82–123.
31. Gärtner, H.-M., and Michaelis, J. A note on the complexity of constraint interaction. In *Logical Aspects of Computational Linguistics, LACL'05*, Lecture Notes in Artificial Intelligence LNCS-3492. Springer, NY, 2005, pp. 114–130.
32. Gärtner, H.-M., and Michaelis, J. Some remarks on locality conditions and minimalist grammars. In *Interfaces + Recursion = Language? Chomsky's Minimalism and the View from Syntax-Semantics*, U. Sauerland and H.-M. Gärtner, Eds. Mouton de Gruyter, NY, 2007, pp. 161–196.
33. Gärtner, H.-M., and Michaelis, J. A note on countercyclicity and minimalist grammars. In *Proceedings of the 8th Conference on Formal Grammar*, G. Penn, Ed. CSLI Publications, Stanford, CA, 2008.
34. Gärtner, H.-M., and Michaelis, J. On the treatment of multiple wh-interrogatives in minimalist grammars. Zentrum für Allgemeine Sprachwissenschaft, Universität Bielefeld, 2010.
35. Ginsburg, S., and Greibach, S. Abstract families of languages. *Memoirs of the American Mathematical Society 87* (1969), 1–32.
36. Goodman, J. Semiring parsing. *Computational Linguistics 25*, 4 (1999), 573–605.
37. Greenberg, J. Some universals of grammar with particular reference to the order of meaningful elements. In *Universals of Language: report of a conference held at Dobbs Ferry, New York, April 13-15, 1961*, J. Greenberg, Ed. MIT Press, Cambridge, Massachusetts, 1963.
38. Guillaumin, M. Conversions between mildly sensitive grammars. UCLA and École Normale Supérieure. http://www.linguistics.ucla.edu/people/stabler/epssw.htm, 2004.
39. Hale, J. *Grammar, Uncertainty, and Sentence Processing*. PhD thesis, Johns Hopkins University, 2003.
40. Harkema, H. A recognizer for minimalist grammars. In *Sixth International Workshop on Parsing Technologies, IWPT'00* (2000). http://www.informatics.susx.ac.uk/research/groups/nlp/carroll/iwpt2000/after.html.
41. Harkema, H. A characterization of minimalist languages. In *Logical Aspects of Computational Linguistics* (NY, 2001), P. de Groote, G. Morrill, and C. Retoré, Eds., Lecture Notes in Artificial Intelligence, No. 2099, Springer, pp. 193–211.
42. Harkema, H. *Parsing Minimalist Languages*. PhD thesis, University of California, Los Angeles, 2001.

43. Hill, J. C., and Wayne, A. A CYK approach to parsing in parallel: A case study. In *Proceedings of The Twenty-Second SIGCSE Technical Symposium on Computer Science Education* (1991), p. 240245.
44. Hiraiwa, K. Movement and derivation: Eliminating the PBC. In *Penn Linguistics Colloquium 26* (2002).
45. Hornstein, N. Movement and control. *Linguistic Inquiry 30* (1999), 69–96.
46. Johnson, M. *Attribute Value Logic and The Theory of Grammar*. No. 16 in CSLI Lecture Notes Series. CSLI Publications, Chicago, 1988.
47. Joshi, A. How much context-sensitivity is necessary for characterizing structural descriptions. In *Natural Language Processing: Theoretical, Computational and Psychological Perspectives*, D. Dowty, L. Karttunen, and A. Zwicky, Eds. Cambridge University Press, NY, 1985, pp. 206–250.
48. Kayne, R. S. *The Antisymmetry of Syntax*. MIT Press, Cambridge, Massachusetts, 1994.
49. Kobele, G. M. Formalizing mirror theory. *Grammars 5* (2002), 177–221.
50. Kobele, G. M. A derivational theory of copying in minimalist grammars. ZAS Syntaxkreis presentation. http://www.linguistics.ucla.edu/people/grads/kobele/papers.htm, 2005.
51. Kobele, G. M. *Generating Copies: An Investigation into Structural Identity in Language and Grammar*. PhD thesis, UCLA, 2006.
52. Kobele, G. M., Retoré, C., and Salvati, S. An automata-theoretic approach to minimalism. In *Model Theoretic Syntax at 10. ESSLLI'07 Workshop Proceedings* (2007), J. Rogers and S. Kepser, Eds.
53. Koopman, H., and Szabolcsi, A. *Verbal Complexes*. MIT Press, Cambridge, Massachusetts, 2000.
54. Koulouris, A., Koziris, N., Andronikos, T., Papakonstantinou, G. K., and Tsanakas, P. A parallel parsing VLSI architecture for arbitrary context free grammars. In *International Conference on Parallel and Distributed Systems, ICPADS'98* (1998), pp. 783–790.
55. Kracht, M. Syntactic codes and grammar refinement. *Journal of Logic, Language and Information 4* (1995), 41–60.
56. Lebeaux, D. Relative clauses, licensing, and the nature of the derivation. In *Perspectives on Phrase Structure: Heads and Licensing*, S. Rothstein, Ed. Academic Press, San Diego, 1991, pp. 209–239.
57. Maneth, S. *Models of Tree Translation*. PhD thesis, Universiteit Leiden, 2004.
58. Marsh, W., and Partee, B. H. How non-context free is variable binding? In *Proceedings of the 3rd West Coast Conference on Formal Linguistics* (Stanford, California, 1984), M. Cobler, S. MacKaye, and M. Wescoat, Eds., Stanford Linguistics Association, pp. 179–190.
59. Mateescu, A., and Salomaa, A. Aspects of classical language theory. In *Handbook of Formal Languages, Volume 1: Word, Language, Grammar*, G. Rozenberg and A. Salomaa, Eds. Springer, NY, 1997, pp. 175–251.
60. May, R. *The Grammar of Quantification*. PhD thesis, Massachusetts Institute of Technology, 1977.
61. McDaniel, D. Partial and multiple wh-movement. *Natural Language and Linguistic Theory 7* (1989), 565–604.
62. Michaelis, J. Derivational minimalism is mildly context-sensitive. In *Proceedings, Logical Aspects of Computational Linguistics, LACL'98* (NY, 1998), Springer, pp. 179–198.
63. Michaelis, J. *On Formal Properties of Minimalist Grammars*. PhD thesis, Universität Potsdam, 2001. *Linguistics in Potsdam 13*, Universitätsbibliothek, Potsdam, Germany.
64. Michaelis, J. Transforming linear context free rewriting systems into minimalist grammars. In *Logical Aspects of Computational Linguistics* (NY, 2001), P. de Groote, G. Morrill, and C. Retoré, Eds., Lecture Notes in Artificial Intelligence, No. 2099, Springer, pp. 228–244.

65. Michaelis, J. Notes on the complexity of complex heads in a minimalist grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks, TAG+6* (2002), pp. 57–65.
66. Michaelis, J. Observations on strict derivational minimalism. *Electronic Notes in Theoretical Computer Science 53* (2004), 192–209. Proceedings of the Joint Meeting of the 6th Conference on Formal Grammar and the 7th Meeting on Mathematics of Language (FGMOL '01), Helsinki, 2001. Available http://www.sciencedirect.com/science/journal/15710661.
67. Michaelis, J., Mönnich, U., and Morawietz, F. On minimalist attribute grammars and macro tree transducers. In *Linguistic Form and its Computation*, C. Rohrer, A. Rossdeutscher, and H. Kamp, Eds. CSLI Publications, Stanford, California, 2001, pp. 287–326.
68. Moschovakis, Y. N. What is an algorithm? In *Mathematics unlimited – 2001 and beyond*, B. Engquist and W. Schmid, Eds. Springer, NY, 2001, pp. 919–936.
69. Müller, G. Incomplete category fronting. SfS report 01-96, Seminar für Sprachwissenschaft, Universität Tübingen, 1996.
70. Müller, G. Shape conservation and remnant movement. Presented at the 30th Conference of the North East Linguistic Society, NELS30, 2000.
71. Muysken, P. Parameterizing the notion 'head'. *Journal of Linguistic Research 2* (1982), 57–75.
72. Nederhof, M.-J., and Satta, G. Probabilistic parsing strategies. In *Proceedings of the 3rd AMAST Workshop on Algebraic Methods in Language Processing (AMiLP 2003)* (Verona, Italy, 2003), pp. 305–314.
73. Nunes, J. Sideward movement. *Linguistic Inquiry 32* (2001), 303–344.
74. Paperno, D. Multiple extraction and minimalist grammars. Ms, UCLA, 2008.
75. Parberry, I. Circuit complexity and feedforward neural networks. In *Mathematical Perspectives on Neural Networks*, P. Smolensky, M. C. Mozer, and D. Rumelhart, Eds. Erlbaum, Mahwah, New Jersey, 1996.
76. Peters, P. S., and Ritchie, R. W. On the generative power of transformational grammar. *Information Sciences 6* (1973), 49–83.
77. Pritchett, B. L. *Grammatical Competence and Parsing Performance*. University of Chicago Press, Chicago, 1992.
78. Richards, N. The principle of minimal compliance. *Linguistic Inquiry 29* (1998), 599–629.
79. Ristad, E. *The Language Complexity Game*. MIT Press, Cambridge, Massachusetts, 1993.
80. Rizzi, L. The fine structure of the left periphery. In *Elements of Grammar*, L. Haegeman, Ed. Kluwer, Boston, 1997, pp. 281–337.
81. Rizzi, L. Relativized minimality effects. In *The Handbook of Contemporary Syntactic Theory*, M. Baltin and C. Collins, Eds. Blackwell, Oxford, 2001, pp. 89–110.
82. Rizzi, L. Locality and left periphery. In *Structures and Beyond: Cartography of Syntactic Structures, Volume 3*, A. Belletti, Ed. Oxford, NY, 2004, pp. 104–131.
83. Rizzi, L. On the form of chains: Criterial positions and ECP effects. In *Wh-Movement: Moving On*, L. L.-S. Cheng and N. Corver, Eds. MIT Press, Cambridge, Massachusetts, 2004, pp. 97–134.
84. Rizzi, L., and Shlonsky, U. Strategies of subject extraction. In *Interfaces + Recursion = Language? Chomsky's Minimalism and the View from Syntax-Semantics*, U. Sauerland and H.-M. Gärtner, Eds. Mouton de Gruyter, NY, 2007, pp. 115–160.
85. Roberts, I. Clitics, head movement, and incorporation. Manuscript, Downing College, University of Cambridge, 2006.
86. Rogers, J. Studies in the logic of trees with applications to grammar formalisms. Tech. Rep. TR-95-04, Department of Computer and Information Sciences, University of Delaware, Newark, Delaware, 1994.

87. Rogers, J. *A Descriptive Approach to Language-Theoretic Complexity*. Cambridge University Press, NY, 1999.
88. Rudin, C. On multiple questions and multiple wh-fronting. *Natural Language and Linguistic Theory 6* (1988), 445–501.
89. Salomaa, A. *Formal Languages*. Academic, NY, 1973.
90. Seki, H., Matsumura, T., Fujii, M., and Kasami, T. On multiple context-free grammars. *Theoretical Computer Science 88* (1991), 191–229.
91. Stabler, E. P. Derivational minimalism. In *Logical Aspects of Computational Linguistics*, C. Retoré, Ed. Springer-Verlag (Lecture Notes in Computer Science 1328), NY, 1997, pp. 68–95.
92. Stabler, E. P. Recognizing head movement. In *Logical Aspects of Computational Linguistics*, P. de Groote, G. Morrill, and C. Retoré, Eds., Lecture Notes in Artificial Intelligence, No. 2099. Springer, NY, 2001, pp. 254–260.
93. Stabler, E. P. Comparing 3 perspectives on head movement. In *From Head Movement and Syntactic Theory, UCLA/Potsdam Working Papers in Linguistics*, A. Mahajan, Ed. UCLA, 2003, pp. 178–198.
94. Stabler, E. P. Varieties of crossing dependencies: Structure dependence and mild context sensitivity. *Cognitive Science 93*, 5 (2004), 699–720.
95. Stabler, E. P. Sidewards without copying. In *Formal Grammar'06, Proceedings of the Conference* (Stanford, 2006), P. Monachesi, G. Penn, G. Satta, and S. Wintner, Eds., CSLI Publications, pp. 133–146.
96. Stabler, E. P. Tupled pregroup grammars. In *Computational Algebraic Approaches to Morphology and Syntax*, C. Casadio and J. Lambek, Eds. Polimetrica, Milan, 2007.
97. Stabler, E. P. A note on implementing relativized minimality. *Forthcoming* (2010).
98. Stabler, E. P., and Keenan, E. L. Structural similarity. *Theoretical Computer Science 293* (2003), 345–363.
99. Starke, M. *Move Dissolves into Merge: A Theory of Locality*. PhD thesis, University of Geneva, Geneva, 2001.
100. Takashi, N., Torisawa, K., Taura, K., and Tsujii, J. A parallel CKY parsing algorithm on large-scale distributed-memory parallel machines. In *Conference of the Pacific Association for Computational Linguistics, PACLING-1997* (1997).
101. Torenvliet, L., and Trautwein, M. A note on the complexity of restricted attribute-value grammars. In *Proceedings of Computational Linguistics In the Netherlands, CLIN5* (Twente, The Netherlands, 1995), M. Moll and A. Nijholt, Eds., Department of Computer Science, University of Twente, pp. 145–164.
102. van Emde Boas, P. Machine models and simulations. In *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed., vol. A. MIT Press, Cambridge, Massachusetts, 1990, pp. 1–66.
103. Vijay-Shanker, K., and Weir, D. The equivalence of four extensions of context free grammar formalisms. *Mathematical Systems Theory 27* (1994), 511–545.