Computational biolinguistics, complexity and the justification of grammars[1]

Pauli Brattico 2024

Abstract: Advanced linguistic theories have an adverse complexity profile in which the number of calculations required to verify them against nontrivial datasets increases exponentially as a function of data complexity. The task implies millions of calculation steps even for regular sentences and soon becomes infeasible for all traditional paper-and-pencil methods. In this article we propose a computational solution to this problem. We formalize a variant of the bottom-up minimalist grammar in a machine-readable notation, embed the result inside a computational infrastructure which mechanizes linguistic reasoning and then consider the use of this construct in the development, justification and testing of linguistic theories and hypotheses in the manner originally proposed by Chomsky (1957).

1 Introduction

Most grammars are inherently combinatorial: they create linguistic expressions by combining primitive parts ("words") into larger units ("phrases"). The grammar is said to be observationally adequate if it generates an expression iff that expression is attested in some (or several, all) languages. The combinatorial nature of grammatical theories presents a problem, however: the deductive chains from initial lexical selections to the actual expressions soon become so protracted

[1] This is a draft (17. 2. 2024) accompanying the source code https://github.com/pajubrat/Template3.

and long as to be virtually impossible to construct and verify by paper-and-pencil methodology.[2] Here we regard this issue as a computational problem and propose a computational solution to it.[3] Section 2 lays down the foundations, Sections 3–4 discuss certain special topics such as the nature of the lexicon, head movement and phrasal movement. Section 5 is reserved for broader discussion. All solutions proposed and discussed in this article are available in the source code repository together with the datasets used in the demonstrations.[4]

2 Derivational search function

Before we can look at justification specifically we need some empirical claims to be justified. The choice is to some extent arbitrary as long as the grammar is presented in a sufficiently rigorous way to make full formalization a feasible approach. Perhaps the simplest grammatical theory available today that satisfies this requirement is the minimalist grammar originally designed by Chomsky (1995) and then developed by the author (Chomsky, 2000, 2001, 2008, 2013) and many others. In this theory linguistic phrase structures are binary sets {X Y}, thus a sentence such as *the dog bites the man* would be represented as {{*the dog*} {*bites* {*the man*}}}. The downside is that linearization as well as subcategorization/selection (and thus labeling/head algorithm) become nontrivial

---

[2] The fairly standard or conservative generative model described later in this article requires f(2) = 2; f(3) = 18; f(4) = 228; f(5) = 4580; f(6) = 137430; f(7) = 5772102 calculation steps as a function of the size of the initial numeration. To translate these numbers into a more concrete linguistic context, calculating that the properties of an expression such as *the man believes that the dog barks* follow from a fairly standard grammatical theory requires approximately 15 million calculation steps. It is not sufficient that these calculations are performed once; they must be executed each time the theory is changed, and in the context of a realistic research program they must be executed against several expressions that bear upon the hypothesis under consideration.

[3] The solution presented in this article is based on earlier ideas first sketched by Chomsky (1957). Chomsky was concerned with a "formalized general theory of linguistic structure" that was applied to linguistic data by comparing the predictions of the theory with observed reality. For example, he observed that by "pushing a precise but inadequate formation to an unacceptable conclusion, we can often expose the exact source of this inadequacy and, consequently, gain a deeper understanding of the linguistic data" (p. 5).

[4] https://github.com/pajubrat/Template3. To install, test and replicate what is documented in this manuscript, first clone this project into a directory in the local machine and write "python Template3.py" into the command prompt when inside the said directory. To run the script the user must have Python (3x) installed on the local machine.

problems that would take us too far from the main topic. Let us assume a less controversial asymmetric binary-branching bare phrase structure [X Y] as the starting point. Having settled on an initial phrase structure formalism we can begin to look at the complexity problem.

We need a computational function that explores all logical implications of the grammar. Let us assume that there is a syntactic working memory (sWM) which contains all syntactic phrase structure objects (henceforth, *syntactic objects*) currently under active consideration. The derivation begins by populating the sWM with zero-level syntactic objects together said form the *numeration*. First we make the simplification that the initial numeration corresponds to the "words" from which the expression are built by using the rules of the grammar.[5] For example, an expression such as *the dog bites the man* can be derived from an initial numeration {*the*, *dog*, *bites*, *the*, *man*} by merging the elements together in a well-defined order. To model this operation in a precise way we posit a recursive *derivational search function* that takes the contents of the syntactic working memory as an input and applies all operations in the grammar (currently only Merge) to all syntactic objects in sWM in a well-ordered sequence, updates the contents of the sWM, and calls the same function recursively with the updated sWM. The derivation ends if only one object is left, after which the result is evaluated and, if it passes as a well-formed output, linearized into a sentence accepted and enumerated by the grammar.[6] The derivational search function is provided below in pseudocode; the computational implementation is in the source code.[7]

[5] We do not reject the possibility of feeding the numeration with complex syntactic objects, although this option will not be considered in this article. Moreover, the notion of primitive syntactic object is neither trivial nor easy to characterize, see Section 3. Another possibility is to populate the numeration with linguistic features which are composed into lexical items by further combinatorial rules. This would make no difference to what we say below.

[6] If we intent to compare the output of the grammar with concrete linguistic expressions, the latter which are typically and here as well conceived as linear strings of word-like objects (say, phonological words), then the theory must have a linearization algorithm. Furthermore, depending on how the internal structure of words is represented we might also need a component for producing linearized morpheme sequences and morpheme boundaries of various types.

[7] https://github.com/pajubrat/Template3/blob/main/Template3.py, function *derivational_search_function*.

(1) *Derivational search function*

a. Assume a set sWM of syntactic objects as input;

b. if there is only one syntactic object, evaluate and print it out;

c. otherwise:

c.1 for each operation O in the grammar:

c.2 for each pair X, Y of items in sWM:

c.2.1 apply O(X, Y) = α;

c.2.2 create updated sWM* which contains α but not X, Y;

c.2.3 call (1) with sWM*.

When applied to four empty words *a*, *b*, *c* and *d* algorithm (1) generates 144 phrase structure representations when it surveys all possible ways of merging them into asymmetric bare phrase structure representations. We can then assume (in the lack of justification for anything more complex) the standard left-to-right depth-first linearization algorithm which turns the 144 phrase structures into linearized sentences. The first three steps of the derivation are shown below:

```
1.

    [a], [b], [c], [d]

    Merge(a, b)
    = [a b]

    [a b], [d], [c]

2.

    [a b], [d], [c]

    Merge([a b], d)
    = [[a b] d]

    [[a b] d], [c]

3.

    [[a b] d], [c]

    Merge([[a b] d], c)
    = [[[a b] d] c]

    [[[a b] d] c]

    |== [[[a b] d] c]   <= ACCEPTED: a b d c
```

Some terminological issues must be clarified before discussing more realistic examples. We want to draw a clear distinction between *computational*, *algorithmic* and *implementation* level descriptions, as originally proposed by Marr (1982). A computational level description is concerned with abstract mappings and ignores both the algorithm and the physical implementation. If we ignore the algorithm (1), what is left is an abstract mapping between a lexical selection and a set of pairs of linearized surface sentences and phrase structure analyses.[8] For example, the above system mapped the numeration {*a*, *b*, *c*, *d*} into 144 ⟨linearized sentence, phrase structure⟩ pairs, of which the first 12 are shown below:

[8] It is not a coincidence that this mapping corresponds to the standard Y-architecture of the minimalist theory, in which lexical choices branch into concrete sentences and syntactic phrase structure representations, in the minimalist parlance to the PF-interface representations (leading into sensorimotoric systems) and LF-interface representations (leading into conceptual-intentional systems).

```
(1) a b d c  [[[a b] d] c]
(2) c a b d  [c[[a b] d]]
(3) d a b c  [d[[a b] c]]
(4) a b c d  [[[a b] c] d]
(5) d a b c  [[d[a b]] c]
(6) c d a b  [c[d[a b]]]
(7) a b d c  [[a b][d c]]
(8) d c a b  [[d c][a b]]
(9) d c a b  [d[c[a b]]]
(10) c a b d  [[c[a b]] d]
(11) c d a b  [[c d][a b]]
(12) a b c d  [[a b][c d]]
```

The same mapping, however, could be generated by an unbounded number of different algorithms, for example, we could reverse-engineer (1) and generate the same mapping by beginning from the surface sentences. We could also consider a variation of (1) that mimics real language processing and/or comprehension. Both approaches could implement the same computational level mapping.[9]

This approach does not depend on whether the theory itself is representational or derivational. A *representational* grammatical theory captures grammaticality by positing well-formedness conditions that apply to complete grammatical representations. The derivational search function then constructs the representations satisfying the well-formedness constraints and tests them against data. A *derivational* grammar differs from the representation grammar in that the construction process itself, and not just the output, is subject to grammatical laws. From the point of view of the derivational search function the difference is relatively inconsequential: both theories generate grammatical objects by rules and in both cases the output is compared with observation. If the grammar is derivational, then at least some properties of the derivation itself are part of the empirical content of the theory.

We must also distinguish enumerative grammars from recognition grammars. An *enumerative grammar*, such as the one just examined, is a procedure which generates a set of expressions;

[9] The implementation level description concerns the ways in which the algorithm is turned into a physical system that performs the actual task. We use the Python programming language for implementation in this article; any general-purpose programming language could do. In the case of the real natural languages the underlying implementation platform is the human brain.

everything else logically derivable from the same lexical items but not derived by the grammar is judged ungrammatical. Most grammatical theories available today are formulated in this way. The derivational search function described above performs the enumeration. A *recognition grammar*, in contrast, is a procedure which decides for any given input expression whether it is grammatical or ungrammatical. Instead of enumerating the set of grammatical expressions it provides a characteristic function for the set. Enumerative and recognition grammars are equivalent in the sense that it is possible to construct one from the other (as long as they compute in finite time). When applied to linguistic theorizing, however, they do have significant differences: recognition grammars are often interpreted implicitly or explicitly as models of "comprehension," while enumerative grammars are conceived as "production" models. This characterization is imprecise and informal, perhaps misleading, but not unimportant when working with empirical linguistic theories.

Testing the derivational search function with empty words is useful in making sure that the algorithm explores the whole derivational space, but unrealistic as a linguistic model. Real words have features, such as subcategorization, which restrict their distribution and in turn relies on a labeling. We begin with the following head/labelling algorithm:

(2) *Labeling/head for any syntactic object α*

a. If α is primitive, it will be the head;

b. suppose α = [X Y], then if X is primitive, it will be the head; otherwise

c. if Y is primitive, it will be the head; otherwise

d. apply (2) recursively to Y.

The output will change into the following:

```
(1) c d a b  [_aP [_cP c d][_aP a b]]
(2) a b c d  [_cP [_aP a b][_cP c d]]
(3) d c a b  [_dP d[_cP c[_aP a b]]]
(4) c a b d  [_dP [_cP c[_aP a b]] d]
(5) d c a b  [_aP [_dP d c][_aP a b]]
(6) a b d c  [_dP [_aP a b][_dP d c]]
(7) d a b c  [_cP [_dP d[_aP a b]] c]
(8) c d a b  [_cP c[_dP d[_aP a b]]]
(9) a b c d  [_dP [_cP [_aP a b] c] d]
(10) d a b c  [_dP d[_cP [_aP a b] c]]
(11) a b d c  [_cP [_dP [_aP a b] d] c]
(12) c a b d  [_cP c[_dP [_aP a b] d]]
```

where the calculated label/head L of complex syntactic object is denoted as [_LP X Y]. Is (2) correct or even plausible? Because we are working with a computational algorithm with the explicit purpose of testing the theory later against data, at this point we can regard (2) as speculation. The important point, instead, is that once we know the head, we can add subcategorization. Let us stipulate two lexical features [!COMP:F] and [–COMP:F] which mandate and ban, respectively, feature F to/from the head of the complement of the lexical item having either feature. Feature F will usually be a major lexical category, but other features are also possible. This grammar will be able to regulate head-complement structures, but cannot rule out ungrammatical specifier-head constructions, so we introduce [!SPEC:F] and [–SPEC:F] for specifier selection with the notion of specifier denoting all left phrases inside the projection from head X. Having introduced subcategorization, we must decide where it applies. There are three solutions: (i) at the stage where grammatical operations apply; (ii) at some intermediate construction ("phase", in the sense of (Chomsky, 2000, 2001)); (iii) at the final output. The first buys us considerable and ultimately necessary savings in computational complexity, so we assume it on such grounds alone; (iii) is also an obvious choice and will later become "interface legibility conditions." Assumption (ii) could be applied if the derivations were broken down into separate sub-derivations, but since this option was not implemented (see however Section 5), this alternative is irrelevant. Assumption (i) requires we block derivations violating subcategorization inside the derivational search function. We add the test to (1), step c.2.1, such that Merge(X, Y) is performed if and only if Y is compatible with the selection features of X.

These assumptions, though still extremely rudimentary, allow us to create and test a simple VP-grammar which enumerates sentences like *the dog bites the man* and *the man bites the dog* with the internal structure [$_{VP}$ [$_{DP}$ *the man*][$_{VP}$ *bites* [$_{DP}$ *the dog*]]] and rules out ungrammatical word permutations. The grammar executes 128 derivational steps and generates the following 12 output sentences:

```
(1) the dog bite the man  [_VP [_DP the dog][_VP bite[_DP the man]]]
(2) the man bite the dog  [_VP [_DP the man][_VP bite[_DP the dog]]]
(3) the dog bite the man  [_VP [_DP the dog][_VP bite[_DP the man]]]
(4) the man bite the dog  [_VP [_DP the man][_VP bite[_DP the dog]]]
(5) the dog bite the man  [_VP [_DP the dog][_VP bite[_DP the man]]]
(6) the man bite the dog  [_VP [_DP the man][_VP bite[_DP the dog]]]
(7) the dog bite the man  [_VP [_DP the dog][_VP bite[_DP the man]]]
(8) the dog bite the man  [_VP [_DP the dog][_VP bite[_DP the man]]]
(9) the man bite the dog  [_VP [_DP the man][_VP bite[_DP the dog]]]
(10) the man bite the dog  [_VP [_DP the man][_VP bite[_DP the dog]]]
(11) the man bite the dog  [_VP [_DP the man][_VP bite[_DP the dog]]]
(12) the dog bite the man  [_VP [_DP the dog][_VP bite[_DP the man]]]
```

The output contains several identical sentences and even identical phrase structures. This is because solutions (1–12) represents different derivations, not different sentences (the mappings from derivations into output phrase structures into linearized sentences are all many-to-one). The following screenshot shows what happened during the first three steps of the derivation:

```
1.

    [the], [man], [bite], [the], [dog]

    Merge(the, man)
    = [_DP the man]

    [the], [dog], [bite], [_DP the man]

2.

    [the], [dog], [bite], [_DP the man]

    Merge(the, dog)
    = [_DP the dog]

    [_DP the man], [_DP the dog], [bite]

3.

    [_DP the man], [_DP the dog], [bite]

    Merge(bite, [_DP the man])
    = [_VP bite[_DP the man]]

    [_DP the dog], [_VP bite[_DP the man]]
```

Since the derivational search function (if written correctly) performs an exhaustive search, sentences (1–12) are the *only* sentences this grammar enumerates from the given numeration: all other permutations and structures are implicitly judged ungrammatical. Thus, the grammar judges **the man the dog bite* ungrammatical, although this fact is not explicitly stated anywhere. However, we haven't constructed any real proof that the grammar is correct – we eyeballed the output and compared it with the intended output we "had in mind." We can fix this issue by providing the derivational search function with a set of target sentences. Moreover, both the numeration and the target sentences can be written into an external input file so that they can be controlled outside of the source code itself. For example, we can write the following three lines into the external input file:

```
Numeration=the,man,bite,the,dog
the man bite the dog
the dog bite the man
```

The first line declares the numeration, the two that follow are interpreted as the target sentences against which the grammar is evaluated. Running the model with these additions shows that the model is observationally adequate. External files like this will become the *datasets* that justify grammatical hypotheses. For example, we can now return to the question of whether the head algorithm (2) is correct. We could assume a variation in which the head algorithm recurses into the left and not to the right and run the model with the same (still trivial) dataset. The new algorithm is still observationally adequate – to our initial surprise. We need additional data to distinguish the two variations of (2). Indeed, in the context of a more realistic research project the dataset will be much more complex than suggested by the simple examples analyzed so far, and involve several numerations paired with the target sentences so that the hypotheses can tested against large batches of data.

While this procedure is able to justify grammars against datasets, it does not determine whether the grammar is "plausible." We could have accomplished the same result by a table-lookup system which pairs input sentences directly with the intended grammaticality judgments. This is not a problem, however, since the output was generated by grammatical representations and operations that (hopefully) had independent justification. But how do we know what the grammar did when it derived the sentences? Some of the screenshots above containing explicit examples of Merge were taken from the *derivational log file* that the model creates at runtime when it derives sentences on the basis of the input in the dataset files. Specifically, every linguistically meaningful step executed by the derivational search function is recorded into the derivational log file. See the actual source code for how to do this in Python.

3 Lexicon, complex words and head chains

The simple VP-grammar tested above relied on an intuitive notion of "word" that formed the basis of both the derivations and the output sentences. Technically these words were provided in their own data structure (call it the *lexicon*) as items defined by a set of lexical features, including the subcategorization features and major lexical categories, and were transformed into primitive syntactic objects that formed the initial numeration (i.e. there was a Lex → SO mapping).[10] This meant that complex words such as *bites* were represented as such in the lexicon; the third person singular suffix and the present tense were represented as lexical features, though these features had no functional rule in the grammar with just one operation. This assumption is not empirically implausible and characterizes a whole category of lexicalist approaches to word formation (Chomsky, 1970; Jackendoff, 1975; Aronoff, 1976; Lapointe, 1980; Anderson, 1982, 1992; Jensen & Stong-Jensen, 1984; Grimshaw & Mester, 1985; Di Sciullo & Williams, 1987; Borer, 1991; Zwicky, 1992; Sells, 1995; Bresnan & Mchombo, 1995; Scalise & Guevara, 2005; Kiparsky, 2017). In such theories the lexical items that form the starting point of syntactic derivations are generated by separate word formation rules. We can either ignore word formation and write the words directly into the lexicon, as we did in Section 2, or add word formation into the derivational search function.

On the other hand, in many generative theories there is a separate syntactic operation which creates complex words by combining primitive syntactic objects. Here we do not make any more specific empirical assumptions about what the domain of this operation is and instead assume for the sake of the example that such an operation exists (as it does exist in many minimalist theories) and that it can create at least some complex words, in particular we will assume that it creates tensed verbs by combining a syntactically less prominent verbal head with a tense head that selects the verb phrase as its complement (Koopman, 1984; Travis, 1984; Baker, 1985, 1988; Pollock, 1989; Borer, 1991; Hale & Keyser, 1993; Roberts, 2001; Julien, 2002: §2; Matushansky, 2006; Dékány, 2018). In schematic terms the operation produces $[_{\mathrm{YP}}\ (_{\mathrm{Y}}\ \mathrm{X}\ \mathrm{Y})^0\ [\ldots\cancel{\mathrm{X}}\ldots]]$ where X is a silent copy of the head that has been copied and adjoined to the higher head Y. It therefore represents a variation of Merge, but with two extra properties: it creates complex zero-level categories instead of

[10] Here it does not matter if the lexical items are sets of lexical features or are represented by more complex structures. If the latter, then the lexical entries would be defined by a custom-made data structure that defines what these more complex entries are.

regular phrases and "copies" something from an existing structure. We add this operation into the theory.

First we posit a feature 'zero-level' to distinguish a complex zero-level category $(_{Y}\ X\ Y)^0$ from a complex phrasal category $[_{\alpha P}\ X\ Y]$ and modify the head algorithm such that it responds to this property and not only to the lack of daughter constituents. At this point we can regard the zero-level property as a stipulation. Then we post an operation *Head Merge* which merges two constituents but creates a zero-level category instead of a regular phrase, the latter which is the output of regular Merge.[11] That is, regular Merge creates $[_{\alpha P}\ X\ Y]$, Head Merge $(X\ Y)^0$.[12] Next we determine how the properties of the newly created complex zero-level categories are calculated on the basis of their constituents. One possibility (the preferred one, in our view) is to change the head algorithm (2) such that it calculates heads also for complex zero-level categories and then let the features of $(X\ Y)^0$ depend on the features of its head, in the sense of Williams (1981), Selkirk (1982) and Di Sciullo & Williams (1987); another is to posit separate feature inheritance inside the complex head. We used the latter since the former requires a notion of adjunction that we currently do not have in the grammar.[13] Feature inheritance was implemented inside Head Merge and copies the features of Y to $(X\ Y)^0$. Let us assume that the morpheme boundaries corresponding to Head Merge are represented by # in the output sentences.

If we just added Head Merge to the list of syntactic operations accessed by the derivational search function, a rather large volume of new derivations would emerge in which the grammar

---

[11] This assumption is not required if we can control the distribution of the 'zero-level' features elsewhere, but having a separate function makes everything explicit and allow us to encapsulate the code that handles the creation of complex words.

[12] Instead of positing HM, we could have relied on regular Merge and posited additional rules and code for classifying some outputs as zero-level objects. At this point, however, it is generally better to posit a specialized function that "encapsulates" the properties of HM into one place, so that we can better manipulate and control its properties. We do not know a priori what properties this operation will have. We can later refactor everything that takes place inside the HM into other components if the result will lead us into a more elegant theory.

[13] Complex heads are created by left-adjunction such that the head of the resulting complex object (X Y) is Y and not X, as it would be if (2) were applied to these objects.

produces new complex words and inserts them into the derivations. Although this is one possible starting point and not empirically impossible if the operation is controlled in some way,[14] this does not match with the original specification $\alpha = [_{YP}\ (_{Y}\ X\ Y)^0\ [_{XP} \ldots \cancel{X} \ldots]]$ which copies X from within the merge partner XP. So far we have assumed that the derivational search function operates with a set of syntactic objects that it combines into new objects; α does exactly this, in that it merges Y with XP, but it contains an additional step where something is copied from within XP.

One possibility is that Head Move is a grammatical operation like Merge and applies freely: α will become the composite operation of Merge + Head Move which occasionally just happens to apply in a sequence but does not need to do so. This presupposes that the objects selected as targets for syntactic operations inside the derivational search function include not only the contents of the syntactic working memory but also the daughters of those elements.[15] This approach maintains full generality and consequently opens up derivations where all syntactic operations can target the internal constituents of the syntactic objects in the syntactic working memory. The hypothesis leads to two issues that ultimately caused us to reject it. First, it introduces an enormous amount of derivational paths that we never see in reality, for example, it becomes possible to perform sideward head movement. This forces us to posit constraints to counterbalance all the illicit derivations, so much so that the restrictions will effectively nullify the whole idea. Another consequence is that the operation is countercyclic: it assumes that X can be inserted inside [Y XP] which requires that we open up previously established phrase structure geometry and then repair it, requiring nontrivial code.[16]

---

[14] Several minimalist models assume an operation of this type, see López (2015) and Embick (2004) and and the literature cited therein.

[15] Either all daughters or daughters selected by some additional criteria. Regardless of the choice, enormous amounts of new derivations will open up.

[16] Specifically, a countercyclic derivation must (i) detach Y from its mother β, (ii) create a new constituent $\alpha = (X\ Y)^0$ by Head Merge and (iii) insert α as the new daughter for β. Operations (i, iii) must be added to the formalism, while (ii) is Head Merge.

A second solution is to posit a special category of "internal" syntactic rules into the derivational search function which targets one syntactic object X and then tampers with its internal properties. Head Move targets some X, locates its head (or all heads) and applies the operation either freely or as a reflex of some triggering condition. This hypothesis still requires countercyclic operations and the special internal grammatical rules, but it generates much less superfluous derivations than the first solution.

The two solutions could be implemented and experimented with, in fact we gave both serious consideration and implemented part of the second approach. However, there is a third alternative: Head Move is part of Merge, in the sense that right before [Y XP] is created Head Move is applied to the head X of XP (thus, in agreement with the Head Movement Constraint (Koopman, 1984; Travis, 1984)) if and only if Y is a zero-level object with a feature making it a bound morpheme. The operation then copies X and merges it with Y by Head Merge introduced earlier, silences the original X phonologically and creates $[_{YP} (X\ Y)^0\ [_{XP} \ldots \cancel{X} \ldots]]$. The operation is cyclic, does not require modifications to phrase structure geometry established before and applies under restricted contexts. For example, all types of sideward head copying are automatically excluded and the operation satisfied HMC. We can perhaps think of Head Move as some kind of "repair" operation executed before Merge.

To test this model against data we posit a new tense head T* into the grammar, and assume it is a bound morpheme and selects for VP. To verify this grammar we assume a simple numeration {*T**, *the*, *dog*, *bark*} where *bark* is an intransitive verb and use the target sentence *the man bark#T** (dataset #3).[17] Running the grammar with these assumptions produces the following:

[17] It is assumed that *bark#T** would be replaced with the vocabulary item *barks*, but this process, which is trivial to implement by brute force, is not useful here because it would "mask" the derivational history of the word.

```
(1) bark#T* the dog  [_TP (bark T*)[_VP [_DP the dog] __ ]]
Derivational steps: 10
Errors 2
Should not generate: {'bark#T* the dog'}
Should generate: {'the dog bark#T*'}
```

From (1) we can verify that the grammar created a complex tensed verb (V, T*)^0 by copying the verb from within the VP. But as it did so, it generates an VS word order that was not among the target sentences, as shown by the fact that the script comes back with two errors. This is because VS sentences are not grammatical in English. The reason this grammar did not put the argument to the preverbal SpecTP position is because we assumed in the lexicon that the intransitive verb must have a DP specifier. If we remove this assumption, the grammar generates the correct word order *the man barks* but with a questionable analysis [$_{TP}$ [DP *the man*] [$_{TP}$ (*bark* T*) __]]:

```
(1) the dog bark#T*  [_TP [_DP the dog][_TP (bark T*) __ ]]
(2) the dog bark#T*  [_TP [_DP the dog][_TP (bark T*) __ ]]
(3) bark#T* the dog  [_TP (bark T*)[_VP [_DP the dog] __ ]]
Derivational steps: 10
Errors 1
Should not generate: {'bark#T* the dog'}
Should generate: set()
```

Although the surface sentence is now correct, the phrase structure analysis is implausible. It is usually assumed that all thematic arguments must be merged inside the VP to receive a thematic role (Fukui & Speas, 1986; Sportiche, 1988; Koopman & Sportiche, 1991), and that the subcategorization features of verbal elements are checked inside the VP.

Transitive verbs are usually analyzed as being created by merging verbal stems with a transitivizer v which introduces the external (agent, causer) argument to its specifier, so that all thematic arguments are initially generated inside the vP where they receive thematic roles and satisfy subcategorization. We introduce a bound morpheme v into the lexicon such that it requires a DP-specifier and selects for a VP. Adding v into the numeration generates 16 sentences after 4586 derivational steps:

```
(1) bite#v#T* the dog the man  [_TP ((bite v) T*)[_vP [_DP the dog][_vP __ [_VP __ [_DP the man]]]]]
(2) bite#v#T* the dog the man  [_TP ((bite v) T*)[_vP [_DP the dog][_vP __ [_VP __ [_DP the man]]]]]
(3) bite#v#T* the man the dog  [_TP ((bite v) T*)[_vP [_DP the man][_vP __ [_VP __ [_DP the dog]]]]]
(4) bite#v#T* the dog the man  [_TP ((bite v) T*)[_vP [_DP the dog][_vP __ [_VP __ [_DP the man]]]]]
```

Notice the familiar “snowball” profile of the three zero-level categories making up the final tensed transitive verb, a consequence of the way head movement was defined above. The subcategorization features of v and V require that the two DPs are merged inside the VP, which again leaves the verb at the first position. Although this configuration is grammatical in some languages (Alexiadou & Anagnostopoulou, 1998), it is ungrammatical in English, a problem we address in the next section. Notice that running the model with an elementary transitive numeration consumed already 4586 derivational steps: this is trivial for a computer but unfeasible for paper-and-pencil methodology.

Because head movement was defined as a local operation, this grammar cannot derive sentences in which heads skip over potential targets or where they combine with lower heads. For example, sentences such as (i) **bite#T the man v the dog* are correctly judged ungrammatical. Also (ii) **v#T the man bite the dog* is underivable: we assumed that head movement is always executed when Merge is executed, hence the merger of v will create $(V, v)^0$ as a grammatical reflex. It follows that $[(X\ Y)^0\ [...\cancel{X}...]]$ occurs if and only if Y selects for XP and is a bound morpheme.

There is one additional fact that we need to pay attention to before discussing the problem with English V-initial sentences. The above experiment shows that, with the exception of the V-initial configurations, the model works as intended. To show that the grammar is adequate in a linguistically interesting sense requires that we test it against many more numerations. For example, if we simulate this grammar with a numeration that lacks v but contains both DP-arguments, it will pass SVO and VSO sentences with analyses $[_{TP}$ T $[_{VP}$ DP $[_{VP}$ V DP]]] and $[_{TP}$ DP [T $[_{VP}$ DP V]]. This happens because we have assumed, tacitly as it were, that v is an “optional” element that can be added freely to the sentence to regulate transitive clauses; if we leave it out, the grammar will either insert both arguments inside the VP or it merges one inside TP. Whether we want this to happen or not, this shows that serious linguistic hypotheses must be checked against several numerations. This is a serious issue because the entire force of the justification will depend on the nature of the dataset(s), but as this thought experiment shows the selection of the relevant numerations remains a free parameter. For example, if our grammar has problems in handling missing elements, as it had above, we could ignore this test experiment (even worse, implicitly) from the dataset to make it look as though our model was successful. We could also boost the success rate by adding repetitive variations that we know the model can solve. Thus, the dataset configuration process must be standardized.

4 Phrasal movement

The final experiment (dataset #4) performed in Section 3 produced verb-initial sentences that are ungrammatical in English. The traditional solution to this problem going back to (Chomsky, 1981, 1982) is to assume that functional heads can have a special nonthematic specifier selection feature, call it EPP, which forces them to have a specifier that is not merged directly (or "externally") to the specific position but is copied ("internally") from within the complement to create α = [$_{YP}$ ZP [$_{YP}$ Y [$_{XP}$ ~~ZP~~ [X WP]]]] in an operation we call *phrasal movement*. Implementing an operation of this type without countercyclic derivations requires we instantiate the operation as a response to the EPP feature right after Merge has created [Y_{EPP} XP]. However, and despite the fact that we will follow this template in this article, the two other solutions introduced in the previous section are possible as well and appear to merit full examination. We could add phrasal movement to the catalog of grammatical operations applied at every derivational step and then allow all operations to target both the syntactic objects in the working memory and their internal constituents, or we could create a special category of rules which affect the internal structure of single syntactic objects. Perhaps the claim that the operations apply "freely" (Chomsky, 2008) refers to an architecture of this type. However, adding this rule directly to Merge suffices to remove the VS(O) sentences from the output. For example, numeration {*T*, *the*, *dog*, *bark*} will produce the following output in which the DP-argument is dislocated to SpecTP as a consequence of Merge due to the EPP-feature at T (dataset #6):

```
(1) the dog bark#T  [_TP [_DP the dog]:1 [_TP (bark T)[_VP __:1  __ ]]]
Derivational steps: 10
Errors 0
```

The part of the derivation leading into the accepted output is as follows.

```
6.

    [T], [the], [dog], [bark]

    Merge(the, dog)
    = [_DP the dog]

    [_DP the dog], [T], [bark]

7.

    [_DP the dog], [T], [bark]

    Merge([_DP the dog], bark)
    = [_VP [_DP the dog] bark]

    [_VP [_DP the dog] bark], [T]

8.

    [_VP [_DP the dog] bark], [T]

    Head Chain (T, bark)
    Chain ((bark T), __:1 )
    Merge(T, [_VP [_DP the dog] bark])
    = [_TP [_DP the dog]:1 [_TP (bark T)[_VP __:1  __ ]]]

    [_TP [_DP the dog]:1 [_TP (bark T)[_VP __:1  __ ]]]

    |== [_TP [_DP the dog]:1 [_TP (bark T)[_VP __:1  __ ]]] <= ACCEPTED: the dog bark#T
```

Both head movement and A-chains are generated during the last step when T is merged above the VP. First, T is combined with the verb to create a finite tensed verb; the resulting complex verb triggers A-chain and copies the grammatical subject to SpecTP. This operation is caused by the EPP feature at T. The verb-initial solution is removed from the output.[18] We can test the model also against transitive clauses, which provides correct outputs, but notice that these mechanisms do not correlate A-movement with any "anchoring" properties such as topicness, agreement or case

[18] EPP-induced A-movement has generated substantial debate in the literature (Fernández-Soriano, 1999; Chomsky, 2000, 2001, 2008; Holmberg, 2000; Miyagawa, 2001, 2010; Bošković, 2002, 2007; Rezac, 2004; Epstein & Seely, 2005; Rackowski & Richards, 2005; Landau, 2007) that we cannot review here, and indeed it is not our purpose to provide an empirically motivated solution to this issue but rather to show how an operation of this type could be added to the derivational search function.

assignment, and we did not restrict the type of phrases that can be targeted: the fact that the grammatical subject was moved was an accidental consequence of the fact that it was merged to SpecvP and thereby formed the required local [T vP] configuration for A-movement. This is an obvious problem, but not an implausible starting point since there are languages, such as Finnish (Holmberg & Nikanne, 2002), in which the EPP feature can be checked by nonsubject topics.

In addition, not all phrasal movement is A-movement. Operator movement observed in English interrogatives such as *which man did the dog bite* __, where *which man* is a special interrogative DP that needs to be dislocated to the left periphery of the interrogative clause, have different properties. Moreover, English interrogativization is associated with Aux-inversion that is explicitly visible in English yes/no questions (*did the dog bite the man?*). Let us first add the auxiliary into the lexicon and assume that it represents T. Then, we assume that the force of the sentence (e.g., interrogative, declarative, imperative) is represented by C subcategorizing for TP and test the model with dataset (#9)

```
Numeration=C,the,man,did,v,bite,the,dog
C the man did bite#v#T the dog
C the dog did bite#v#T the man
```

which derives

```
(1) C the man did bite#v the dog  [_CP C[_TP [_DP the man]:1 [_TP did[_VP __:1 [_VP (bite v)[_VP __ [_DP the dog]]]]]]
(2) C the dog did bite#v the man  [_CP C[_TP [_DP the dog]:1 [_TP did[_VP __:1 [_VP (bite v)[_VP __ [_DP the man]]]]]]
(3) C the dog did bite#v the man  [_CP C[_TP [_DP the dog]:1 [_TP did[_VP __:1 [_VP (bite v)[_VP __ [_DP the man]]]]]]
(4) C the dog did bite#v the man  [_CP C[_TP [_DP the dog]:1 [_TP did[_VP __:1 [_VP (bite v)[_VP __ [_DP the man]]]]]]
```

and 12 other sentences, all generating one of the target sentences. Since tense is expressed by *did*, the verb remains at a lower position. All we have to do to model English Aux-inversion is to assume that an interrogative C is a bound item, which generates without any further assumptions the Aux-inversion pattern (#10)[19]:

---

[19] Notice that if we remove the auxiliary, this model produces verb-inversion (*bites#v#T#C the dog the man?*) which is ungrammatical in English but grammatical in many other languages, such as Italian and Finnish, so the derivation is not completely implausible, though something must block it in English.

```
(1) did#C(wh) the man bite#v the dog  [_CP (did C(wh))[_TP [_DP the man]:1 [_TP __ [_VP __:1 [_VP (bite v)[_VP __ [_DP the dog]]]]]]
(2) did#C(wh) the dog bite#v the man  [_CP (did C(wh))[_TP [_DP the dog]:2 [_TP __ [_VP __:2 [_VP (bite v)[_VP __ [_DP the man]]]]]]
(3) did#C(wh) the dog bite#v the man  [_CP (did C(wh))[_TP [_DP the dog]:1 [_TP __ [_VP __:1 [_VP (bite v)[_VP __ [_DP the man]]]]]]
(4) did#C(wh) the dog bite#v the man  [_CP (did C(wh))[_TP [_DP the dog]:1 [_TP __ [_VP __:1 [_VP (bite v)[_VP __ [_DP the man]]]]]]
```

Having these mechanisms in the model, we can model interrogativization. The numeration will be {*C(wh)*, *the*, *man*, *did*, *v*, *bite*, *which*, *dog*} and the target sentences as follows:

```
which dog did#C(wh) the man bite#v
which man did#C(wh) the dog bite#v
which dog did#C(wh) bite#v the man
which man did#C(wh) bite#v the dog
```

Let us assume that C(wh) has an interrogative feature WH which triggers the search for the corresponding interrogative operator that we assume to be D with feature WH (=*which*). Let us assume, furthermore, that after Merge has created [X YP], X is checked for the interrogative feature WH and, if the feature is present, copy and merge is attempted. The search algorithm, call it *minimal search* after Chomsky (2008), moves downstream following the projectional spine of each head H and then enters the complement of H if any, returning first constituent with a WH-feature at its head. This grammar enumerates the following outputs (among others)(dataset #12)

```
(1) which dog did#C(wh) bite#v the man  [_CP [_DP which dog]:3 [_CP (did C(wh))[_TP __:3 [_TP __ [_VP __:3 [_VP (bite v)[_VP __ [_DP the man]]]]]]]
(2) which dog did#C(wh) the man bite#v  [_CP [_DP which dog]:4 [_CP (did C(wh))[_TP [_DP the man]:3 [_TP __ [_VP __:3 [_VP (bite v)[_VP __ __:4 ]]]]]]
(3) which dog did#C(wh) bite#v the man  [_CP [_DP which dog]:3 [_CP (did C(wh))[_TP __:3 [_TP __ [_VP __:3 [_VP (bite v)[_VP __ [_DP the man]]]]]]]
(4) which dog did#C(wh) bite#v the man  [_CP [_DP which dog]:1 [_CP (did C(wh))[_TP __:1 [_TP __ [_VP __:1 [_VP (bite v)[_VP __ [_DP the man]]]]]]]
```

which is the result we were looking for. Both the external argument and the internal argument can be moved to SpecCP, depending on where the interrogative operator *which* is. Because minimal search follows labeling and head-complement configurations, we can already suspect that the CED-effects (Huang, 1982b, 1982a) follow in the sense that left phrases can never be searched. However, the numeration posited above does not *show* this, since all operators fell to the minimal search paths. To show that the CED effects are captured we must craft a numeration from a sentence such as *the dog from which city barks* and show that **which city the dog from __ barks* is not in the output. First, adding the preposition *from* into the grammar with reasonable subcategorization features and testing it with an intransitive numeration together with C and T (dataset #13) produced 276 accepted derivations (several overlaps), among them the irrelevant solutions (3)a–b and the relevant analyses (3)c–d.

(3) a. C the dog barks from the city.

b. From the city C the dog barks (PP base-generated to SpecCP)

c. C the city from the dog barks.

d. C the dog from the city barks.

Once we know that the relevant analysis (3)d is in the output, we can replace the second definite article with *which* and the declarative C with an interrogative C(wh) and run the model to verify that **which city the dog from __ barks* is not among the output (#14). This proves that the grammar does not allow extraction from subjects (extraction from adjuncts cannot be tested since we do not yet have adjuncts in the grammar). However, the experiment again generated a lot of ungrammatical sentences such as those in (4) which show that the grammar far from being even observationally adequate.

(4) a. [$_{PP}$ from the dog [$_{CP}$ which city$_1$ C(wh) $\_\__1$ barks]] ("relative construction")

b. From which city C(wh) barks the dog? (base-generated PP to SpecCP)

c. [$_{CP}$ From the dog [$_{CP}$ which city C __ barks]]? (Double-filled SpecCP)

d. Barks$_1$ the dog $\_\__1$ from which city? (cf. 'Does the man bark from which city?')

The reader can go through the 84 solutions and consider various options for filtering the unwanted outputs. It is also worth noting that these assumptions generate successive-cyclic A-movement provided that there is a sequence of local heads with the EPP feature. For example, the numeration {*the*, *dog*, *T*, *seem*, *to*, *bark*} generates the following output if the raising verb and the infinitival *to* have the EPP feature (dataset #15).

```
(1) the dog seem#T to bark  [_TP [_DP the dog]:1 [_TP (seem T)[_VP __:1 [_VP __ [_T/infP __:1 [_T/infP to[_VP __:1  bark]]]]]]
Derivational steps: 481
Errors 0
```

This facts makes it possible to model English personal passives. If we assume that there exists a special "passive" v* which has an unthematic specifier position (EPP) instead of the external agent argument, we get sentences in which the direct object is moved successive-cyclically to the SpecTP position:

```
(1) the man was bite#v*  [_TP [_DP the man]:2 [_TP was[_VP __:2 [_VP (bite v*)[_VP __ __:2 ]]]]]
Derivational steps: 58
Errors 0
```

The numeration contains *was* = T and the special passive v*, (bite v*)^0 would be spelled out as *bitten*. The direct object is raised successive-cyclically from the VP into Specv*P and then to SpecTP. Notice that we must assume that A-chains can target not only specifiers but also complements. The key portion of the derivation is the following:

```
57.

    [was], [v*], [_VP bite[_DP the man]]

    Head Chain (v*, bite)
    Chain ((bite v*), __:2 )
    Merge(v*, [_VP bite[_DP the man]])
    = [_VP [_DP the man]:2 [_VP (bite v*)[_VP __ __:2 ]]]

    [was], [_VP [_DP the man]:2 [_VP (bite v*)[_VP __ __:2 ]]]

58.

    [was], [_VP [_DP the man]:2 [_VP (bite v*)[_VP __ __:2 ]]]

    Chain (was, __:2 )
    Merge(was, [_VP [_DP the man]:2 [_VP (bite v*)[_VP __ __:2 ]]])
    = [_TP [_DP the man]:2 [_TP was[_VP __:2 [_VP (bite v*)[_VP __ __:2 ]]]]]

    [_TP [_DP the man]:2 [_TP was[_VP __:2 [_VP (bite v*)[_VP __ __:2 ]]]]]

    |== [_TP [_DP the man]:2 [_TP was[_VP __:2 [_VP (bite v*)[_VP __ __:2 ]]]]] <= ACCEPTED: the man was bite#v*
```

First the passive v* is merged, which triggers both head movement and phrasal A-movement and derives the extended verb phrase [$_{\text{v*P}}$ *the man*$_1$ [$_{\text{v*P}}$ (*bite* v*) [$_{\text{VP}}$ ~~*bite*~~ __$_1$]]] (step 57) and which is then followed by A-movement to SpecTP (step 58).

5 Discussion

The source code together with the dataset containing all experiments (#1–15) are available in the source code repository. It is written in the form of one relatively short Python script containing less than 500 lines of commented code that can be modified and expanded as required by the theory and data. Here we consider some of the most important extensions.

Short Python programs can be provided in the form of simple scripts contained in one text file. Since the template used in the present article was designed to serve as a starting point for larger projects, it too is contained in just one file. The file contains definitions for data structures (e.g., lexicon, phrase structure) and functions (e.g., the derivational search function). The script reads the dataset file (specified inside the script) and executes all the experiments #1–16 reported above. Larger projects, however, are typically not organized into one script file but are instead dissolved into several files called *modules* that contain code designed for specific tasks. For example, the derivational search function maps initial numerations into phrase structure representations, the latter which are processed by one additional function generating the sentence/phrase structure -pairs for accepted solutions. These two functions correspond to the three branches of the Y-architecture: the first corresponds to the horizontal branch while the second corresponds to the two vertical branches. In a linguistically realistic model the syntactic objects SO constituting the end point of the derivation would be send to two separate modules, the first which handles mapping into PF and beyond and contains whatever postsyntactic operations are required to calculate the data (Marantz, 1984; Embick & Noyer, 2001; Hale & Keyser, 2002; Matushansky, 2006; Harizanov, 2018) and a second module which maps SO to the LF-interface that feeds a separate module that handles semantic interpretation. The SO-PF mapping will generate formal-morphological surface modifications that are not visible for semantic interpretation, while the SO-LF mapping generates covert syntax. Thus, what in the template used in this study is expressed by two functions will in a more realistic system be implemented by three separate modules. These modules can be brought under the same overall architecture by creating one higher-level “speaker model” container class which maintains the separate modules, so that instantiating objects from the higher-level container class will automatically create instances of the module classes. Items from the numeration can be feed to the container class which manages all data processing in and out of the modules and thus defines the whole architecture.

In addition to Merge and Move, the standard minimalist theory posits a third operation Agree that is responsible for various feature covariance dependencies, such as standard subject-verb agreement. This operation was not implemented; instead, the code already contains the kernel of an operation of this type. Recall that we assumed that C(wh) searches for a corresponding operator element inside its complement before phrasal movement is executed. This was implemented by a minimal search function which takes a wh-feature as input and locates a head that has the same feature. The operation resembles long-distance Agree (or probe-goal dependency). A similar assumption was made tacitly in connection with A-movement, which located the phrase from the complement but did not restrict the nature of the selected element; consequently, if we want to search for elements of specific type, such as grammatical subjects, a feature-based search function is an obvious candidate. It is possible that a theory of Agree could be developed from these mechanisms without positing a separate operation. This approach would, however, rule out upward-directed agreement systems (Chomsky, 1993; Koopman, 2006; Chandra, 2007; Baker, 2008; Merchant, 2011; Zeijlstra, 2012; Carstens, 2016; Bjorkman & Zeijlstra, 2019; Baker & Camargo Souza, 2020; Keine & Dash, 2022) and is therefore neither empirically inconsequential nor something that could be assumed without giving it careful consideration.

We pointed out in connection with several examples that derivations which are possible in many other languages are not possible in English. For example, the current grammar can form yes/no questions by fronting finite verbs, which is not possible in English. To model language-specific crosslinguistic variation we need a theory of the said variation and some technical implementation for "different grammars." The easiest way to accomplish a system of this type is to create several language-specific speaker models and then use them on the basis of the language in the numeration and/or input sentence. Each speaker model would contain a different lexicon (depending on the language) plus different grammar rules and/or parameters, depending on how the variation is modeled. If it is modelled by positing different rules (implausible, but theoretically possible), then the list of grammatical rules for each speaker model should be populated from a larger pool of possible rules when the speaker model instances are generated; if the rules differ only in terms of finite number of parameters, then each speaker model would contain a data-structure holding the values of these parameters in addition to the rules. If languages differ only in terms of their lexicon, then the solution is to pair each speaker model with a different lexicon depending on the language.

The derivational search function in and itself is not a model of an actual speaker/hearer, but part of the executive layer responsible for running the grammar. It can be expanded easily into a more realistic model, however. Suppose we want to create a realistic model for language production and assume that the endpoint of the derivation as depicted in this article constitutes a "motoric plan" which guides speech production. If we assume that the step-by-step derivation is a realistic description of the processed which build up plans for externalized speech, then we can add psycholinguistic plausibility functions to the derivational search function determining the order at which the various elements are merged together and associate each step in the derivation with a cognitive cost. The theory will now make precise predictions of cognitive costs associated with the production of sentences.[20] We can also model and thus predict the order in which real speaker create sentences from lists of words. However, it is far from trivial that a bottom-up model of this type has a measurable connection to language production. It seems to describe some type of abstract "logic" or characteristics that linguistic representations must have.

Let us consider the complexity profiles of the grammars examined in the previous section. The amount of derivational steps required to calculate the logical implications of sentences that contain unrestricted empty words, which define the worst case scenario, is enormous. Thus, while calculating the derivations for four words is 228, seven words require almost six million steps. The

---

[20] A theory of this type requires, however, an additional component which specifies the meaning of the intended sentence so that the derivation can converge towards the target.

number of steps increases exponentially and soon becomes intractable even for a computer. Subcategorization and other restrictions can be used to close off derivational branching. The range of calculations required for simple T-VP-grammars was around several thousands, but transitive clauses raised the number of tens of thousands, with the last experiment involving an interrogative clause with a PP argument calculates through more than one hundred thousand derivations. Although these numbers can most likely be reduced by adding derivational constraints, and are easily managed by modern computers, the growth function is exponential and will, eventually, hit a wall. For example, testing the model with *the man believes that the dog barks* calculated through 15 million steps. A possible solution is to break the initial numeration into sub-numerations ("phases," Chomsky 2000) which are derived independently, and then connect the outputs together in a separate step. Thus, instead of providing the derivational search function with an initial numeration it is provided with a set of numerations which puts restrictions on which items can be merged with which items. This has many consequences for movement operations, but the exact predictions will depend on what the phases are. This method differs from the one used here in that we restrict the combinatorial possibilities before the derivation begins; it is easy to imagine several schemes which accomplish the same output.

As pointed out earlier, using this method for the justification of serious and interesting linguistic hypotheses requires that the hypotheses are tested against several numeration/target sentence pairs. This type of batch testing is easy to implement technically by writing the tests into the one external input file, as was done here, but involves additional concerns since the selection of the input data is still free. One solution is to develop standardized datasets. For example, we could imagine a range of datapoints that any theory of say English passivization has to deduce correctly and then develop of a standardized batch dataset that expresses these properties.

6 Conclusions

Grammatical theories and linguistic hypotheses are logically deep in the sense that the chain of reasoning connecting them with data are long, especially when the data is nontrivial. This renders traditional paper-and-pencil techniques almost completely useless. A rigorous computational methodology was suggested that calculates through all logical implications of the theory and compares the results automatically with data.

References

Alexiadou, A., & Anagnostopoulou, E. (1998). Parametrizing AGR: Word Order, V-movement and EPP Checking. *Natural Language and Linguistic Theory.*, *16*(3), 491–539. https://doi.org/10.1023/a:1006090432389

Anderson, S. R. (1982). Where's Morphology? *Linguistic Inquiry*, *4*, 571–612.

Anderson, S. R. (1992). *A-morphous morphology*. Cambridge University Press.

Aronoff, M. (1976). *Word formation in generative grammar*. Cambridge, MA.: MIT Press.

Baker, M. (1985). The Mirror Principle and Morphosyntactic Explanation. *Linguistic Inquiry*, *16*, 373–415.

Baker, M. (1988). *Incorporation. A theory of grammatical function changing*. University of Chicago Press.

Baker, M. (2008). The Syntax of Agreement and Concord. In *The Syntax of Agreement and Concord*. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511619830

Baker, M., & Camargo Souza, L. (2020). Agree without Agreement: Switch-reference and reflexive voice in two Panoan languages. *Natural Language and Linguistic Theory*, *38*(4), 1053–1114. https://doi.org/10.1007/S11049-019-09463-W/METRICS

Bjorkman, B. M., & Zeijlstra, H. (2019). Checking Up on (ϕ-)Agree. *Linguistic Inquiry*, *50*(3), 527–569. https://doi.org/10.1162/ling_a_00319

Borer, H. (1991). The causative-inchoative alternation: A case study in parallel morphology. *Linguistic Review*, *8*, 119–1158.

Bošković, Ž. (2002). A–movement and the EPP. *Syntax*, *5*(3), 167–218. https://doi.org/10.1111/1467-9612.00051

Bošković, Ž. (2007). On the Locality and Motivation of Move and Agree: An Even More Minimal Theory. *Linguistic Inquiry*, *38*, 589–644.

Bresnan, J., & Mchombo, S. A. (1995). The lexical integrity principle: Evidence from Bantu. *Natural Language and Linguistic Theory*, *13*(2), 181–254. https://doi.org/10.1007/BF00992782

Carstens, V. (2016). Delayed Valuation: A Reanalysis of Goal Features, “Upward” Complementizer Agreement, and the Mechanics of Case. *Syntax*, *19*(1), 1–42. https://doi.org/10.1111/SYNT.12116

Chandra, P. (2007). *Dis(Agree): Movement and agreement reconsidered* [Ph.D. thesis]. University of Maryland.

Chomsky, N. (1957). *Syntactic Structures*. The Hague: Mouton.

Chomsky, N. (1970). Remarks on Nominalization. In R. A. Jacobs & P. S. Rosenbaum (Eds.), *Readings in English Trasformational Grammar*. Wiley.

Chomsky, N. (1981). *Lectures in Government and Binding: The Pisa Lectures*. Dordrecht: Foris.

Chomsky, N. (1982). *Some concepts and consequences of the theory of Government and Binding*. MIT Press.

Chomsky, N. (1993). A minimalist program for linguistic theory. In K. Hale & S. J. Keiser (Eds.), *The view from building 20*. Cambridge, MA.: MIT Press.

Chomsky, N. (1995). *The Minimalist Program*. Cambridge, MA.: MIT Press.

Chomsky, N. (2000). Minimalist Inquiries: The Framework. In R. Martin, D. Michaels, & J. Uriagereka (Eds.), *Step by Step: Essays on Minimalist Syntax in Honor of Howard Lasnik* (pp. 89–156). Cambridge, MA.: MIT Press.

Chomsky, N. (2001). Derivation by Phase. In M. Kenstowicz (Ed.), *Ken Hale: A Life in Language* (pp. 1–37). Cambridge, MA.: MIT Press.

Chomsky, N. (2008). On Phases. In C. Otero, R. Freidin, & M.-L. Zubizarreta (Eds.), *Foundational Issues in Linguistic Theory: Essays in Honor of Jean-Roger Vergnaud* (pp. 133–166). Cambridge, MA.: MIT Press.

Chomsky, N. (2013). Problems of projection. *Lingua*, *130*, 33–49.

Dékány, É. (2018). Approaches to head movement: A critical assessment. *Glossa: A Journal of General Linguistics*, *3*(1). https://doi.org/10.5334/gjgl.316

Di Sciullo, A. M., & Williams, E. (1987). *On the definition of the Word*. Cambridge, MA.: MIT Press.

Embick, D. (2004). On the structure of resultative participles in English. *Linguistic Inquiry*, *35*, 355–392.

Embick, D., & Noyer, R. (2001). Movement operations after syntax. *Linguistic Inquiry*, *32*(4), 555–595. https://doi.org/10.1162/002438901753373005

Epstein, S., & Seely, D. (2005). *Transformations and Derivations*. Cambridge: Cambridge University Press.

Fernández-Soriano, O. (1999). Two Types of Impersonal Sentences in Spanish: Locative and Dative Subjects. *Syntax*, *2*(2), 101–140. https://doi.org/10.1111/1467-9612.00017

Fukui, N., & Speas, M. (1986). Specifiers and projection. *MIT Working Papers in Linguistics*, *8*, 128–172.

Grimshaw, J., & Mester, R.-A. (1985). Complex verb formation in Eskimo. *Natural Language & Linguistic Theory*, *3*(1), 1–19.

Hale, K., & Keyser, S. J. (1993). On Argument Structure and the Lexical Expression of Syntactic Relations. In K. Hale & S. J. Keyser (Eds.), *The View from Building 20: Essays in Linguistics in Honor of Sylvain Bromberger* (pp. 53–109). MIT Press.

Hale, K., & Keyser, S. J. (2002). *Prolegomenon to a theory of argument structure*. Cambridge, MA.: MIT Press.

Harizanov, B. (2018). Word Formation at the Syntax-Morphology Interface: Denominal Adjectives in Bulgarian. *Linguistic Inquiry*, *49*(2), 283–333. https://doi.org/10.1162/LING_a_00274

Holmberg, A. (2000). Scandinavian Stylistic Fronting: How any Category can become an Expletive. *Linguistic Inquiry.*, *31*(3), 445–484. https://doi.org/10.1162/002438900554406

Holmberg, A., & Nikanne, U. (2002). Expletives, subjects and topics in Finnish. In P. Svenonius (Ed.), *Subjects, Expletives, and the EPP* (pp. 71–106). Oxford: Oxford University Press.

Huang, C.-T. J. (1982a). *Logical relations in Chinese and the theory of grammar*. New York & London: Garland.

Huang, C.-T. J. (1982b). Move wh in a language without wh movement. *Linguistic Review*, *1*, 369–416.

Jackendoff, R. (1975). Morphological and Semantic Regularities in the Lexicon. *Language*, *51*(3), 639. https://doi.org/10.2307/412891

Jensen, J. T., & Stong-Jensen, M. (1984). Morphology is in the lexicon! *Linguistic Inquiry*, *15*(3), 474–498.

Julien, M. (2002). *Syntactic Heads and Word Formation*. Oxford: Oxford University Press.

Keine, S., & Dash, B. (2022). Movement and cyclic Agree. *Natural Language and Linguistic Theory*, 1–54. https://doi.org/10.1007/S11049-022-09538-1/METRICS

Kiparsky, P. (2017). Nominal verbs and transitive nouns: Vindicating lexicalism. In C. Bowern, L. Horn, & R. Zanuttini (Eds.), *On looking into words (and beyond)* (pp. 311–345). Berlin: Language Science Press.

Koopman, H. (1984). *The Syntax of Verbs: from Verb Movement Rules in the Kru Languages to Universal Grammar*. Foris.

Koopman, H. (2006). *Agreement configurations: In defense of the "Spec head"* (pp. 159–200). John Benjamins.

Koopman, H., & Sportiche, D. (1991). The position of subjects. *Lingua*, *85*(2/3), 211–258.

Landau, I. (2007). EPP Extensions. *Linguistic Inquiry*, *38*, 485–523.

Lapointe, S. (1980). *A theory of grammatical agreement.* [Ph.D. dissertation]. University of Massachusetts-Amherst.

López, L. (2015). Parallel Computation in Word Formation. *Linguistic Inquiry*, *46*(3), 657–701.

Marantz, A. (1984). *On the Nature of Grammatical Relations*. MIT Press.

Matushansky, O. (2006). Head Movement in Linguistic Theory. *Linguistic Inquiry*, *37*, 69–109.

Merchant, J. (2011). Aleut case matters. In *Pragmatics and autolexical grammar: In honor of Jerry Sadock* (pp. 193–210). Amsterdam: John Benjamins. https://doi.org/10.1075/LA.176.12MER

Miyagawa, S. (2001). Scrambling, EPP, and wh-in situ. In M. Kenstowicz (Ed.), *Ken Hale: A Life in Language* (pp. 293–338). MIT Press.

Miyagawa, S. (2010). *Why Agree? Why Move? Unifying Agreement-Based and Discourse-Configurational Languages*. Cambridge, MA.: MIT Press.

Pollock, J. Y. (1989). Verb Movement, Universal Grammar and the structure of IP. *Linguistic Inquiry*, *20*, 365–424.

Rackowski, A., & Richards, N. (2005). Phase Edge and Extraction: A Tagalog Case Study. *Linguistic Inquiry*, *36*(4), 565–599. https://doi.org/10.1162/002438905774464368

Rezac, M. (2004). The EPP in Breton: An unvalued categorial feature. In H. van Riemsdijk, H. van der Hulst, & J. Koster (Eds.), *Triggers* (pp. 451–492). Berlin: Mouton de Gruyter.

Roberts, I. (2001). Head movement. In M. Baltin & C. Collins (Eds.), *Handbook of syntactic theory* (pp. 113–147). Oxford: Blackwell.

Scalise, S., & Guevara, E. (2005). The lexicalist approach to word-formation and the notion of the lexicon. In P. Stekauer & R. Lieber (Eds.), *Handbook of word-formation* (pp. 147–187). Dordrecht: Springer.

Selkirk, E. (1982). *The Syntax of Words*. Cambridge, MA.: MIT Press.

Sells, P. (1995). Korean and Japanese Morphology from a Lexical Perspective. *Linguistic Inquiry*, *26*(2), 277–325.

Sportiche, D. (1988). A Theory of Floating Quantifiers and its Corollaries for Constituent Structure. *Linguistic Inquiry*, *19*(3), 425–449.

Travis, L. (1984). *Parameters and Effects of Word Order Variation* [Ph.D. thesis]. MIT.

Williams, E. (1981). On the Notions "Lexically Related" and "Head of a Word." *Linguistic Inquiry*, *12*, 245–274.

Zeijlstra, H. (2012). There is only one way to agree. *Linguistic Review*, *29*(3), 491–539. https://doi.org/10.1515/TLR-2012-0017

Zwicky, A. M. (1992). Some choices in the theory of morphology. In *Formal Grammar: Theory and Implementation* (Levine, R., pp. 327–371). Vancouver: Oxford University Press.