Computational biolinguistics, complexity and the justification of grammars

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 (with the lexicon, bare phrase structure, labeling algorithm, Merge, head movement, A-chains, Ā-chains, adjuncts/adjunction, minimal search, linearization, subcategorization, interface legibility conditions) in a machine-readable notation, embed the result inside a computational infrastructure which mechanizes parts of 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). The results show that although the proposed solution solves the problem with purely manual approaches, additional strategies might be needed to bring complexity down to a feasible level when working with very nontrivial data.

1 Introduction

Most grammars are inherently combinatorial: they create linguistic expressions by combining primitive parts ("words") into larger units ("phrases").[1] The grammar is said to be observationally adequate if it generates attested expressions and only them. The combinatorial nature of

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

grammatical theories presents a problem, however: the deductive chains from initial lexical selections to whole expressions soon become so protracted and long as to be virtually impossible to construct and verify by any paper-and-pencil methodology.[2] We propose a computational solution to this problem, 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). The model described in this article and implemented in the source code is a "formalized general theory of linguistic structure" in this precise sense.[3]

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.

---

[2] The number of calculations increases exponentially as a function of the size of the initial lexical set. The number of binary trees for N elements is provided by the Catalan number ((2*n*)!/(*n*!(*n*+1)!), https://oeis.org/A000108), but the amount of calculations is even higher since we must also distinguish different way of generating each tree. To get a rough idea of what this means in a realistic linguistic context, calculating that the properties of an expression such as *the man believes that the dog barked* follow from a fairly standard grammatical theory requires approximately 15 million calculation steps, approximately 30 minutes with the algorithm discussed in this paper. 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 script can be cloned or downloaded from 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.

While we selected this grammar as a starting point, the arguments presented here in no way depend on the particulars of this theory.

Next we construct a computational function that explores all logical implications of the grammar and compares the output with empirical observation. 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. This initial set of zero-level objects will be called *numeration*. First we make the simplification that the initial numeration contains the "words" from which the expression are built by using the rules of the grammar.[4] For example, an expression such as *the dog bites the man* is derived from an initial numeration {*the*, *dog*, *bites*, *the*, *man*} by merging the elements together in some 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 input and applies all operations in the grammar 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. The derivational search function is provided below in pseudocode; the Python implementation is in the source code.

(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;

[4] 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 then composed into lexical items by further combinatorial rules. This would make no difference to what we say below.

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* where α replaces X and Y;

c.2.3 recurse into (1) with sWM*.

To illustrate, let us assume that the grammar contains only one operation Merge which combines syntactic objects X and Y and yields a complex object [X Y].[5] When applied to four empty words *a*, *b*, *c* and *d* the algorithm (1) generates 144 phrase structure representations. The first three steps of the derivation are shown below (actual output from the implemented algorithm (1)):

[5] In some minimalist grammars phrase structures are binary sets {X Y} instead of asymmetric [X Y] assumed in the main text. The downside is that linearization as well as subcategorization/selection (and thus labeling/head algorithm) become nontrivial problems that we do not know at present how to solve and which would take us too far from the main topic. The source code repository contains a variation of the main script which uses the set-theoretical phrase structure formalism (https://github.com/pajubrat/Template1) but does not have the linearization or labeling functions.

```
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
```

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, in the above example the final linearized output sentence was *a b d c*. It was "accepted" by the grammar since we currently have only one operation, Merge, in the theory.

Some clarifications and terminological notes are in order before discussing more realistic examples. The form of the derivational search function does not depend on the particulars of any grammatical theory, even less on minimalism. It is defined as a recursive operation which exhausts the logical implications of a grammatical theory provided only that the theory builds expressions from lexical items ("words") by applying structure-building rules to them. The theory must be sufficiently precise to be amendable for formalization. Once these conditions are satisfied, the theory can be provided with a derivational search function.

We also 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 a mapping between a lexical selection and a set of pairs of linearized surface sentences and phrase structure analyses exhausting the logical content of the theory.[6] 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:

```
(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]]
```

This is what the theory claims to match with empirical reality. 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.[7] The main point, however, is that we do not to confuse algorithms with the computational level descriptions. A useful rule of thumb is that only computational level descriptions – input/output behavior of the algorithm – are evaluated for empirical correctness when the main focus is on linguistic competence; the algorithm, if it proposed as a realistic model, can be evaluated against additional performance data, but that is

[6] 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).

[7] 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.

optional. Moreover, everything described in this article was implemented in Python, a general-purpose programming language particularly well suited for this type of task, but the actual Python code – although it implements the mapping and what is described in (1) – is only one possible implementation and as such introduces many additional properties and details (language syntax, data-structures, class definitions) that are irrelevant to linguistics or to the argument developed in this article.

This approach does not depend on whether the grammar itself is representational or derivational. A *representational* grammar captures grammaticality by positing well-formedness conditions, such as the "interface legibility conditions" of the standard minimalist theory, that apply to complete grammatical representations. The derivational search function then constructs the representations by applying the rules of the grammar to the lexical base and tests the outputs against data. In other words, a representational grammar needs a derivational search function to justify it against observation; the derivational search function is not part of the theory, but rather part of the higher-level infrastructure that exhausts the logical implications of the grammar. A *derivational* grammar differs from the representation grammar in that the construction process itself, and not just the output, is subject to grammatical laws and is therefore subject to empirical argumentation. For example, some minimalist grammars make empirical claims on the basis of the computational resources consumed by the derivations (such models seem to be less prominent today than they once were). Nevertheless, from the point of view of the derivational search function the difference between representation and derivational explanations 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;

everything else logically derivable from the same lexical items but not derived by the grammar is automatically 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. Thus, instead of enumerating the set of grammatical expressions it provides the characteristic function for the same set. Enumerative and recognition grammars are mathematically equivalent in the sense that it is possible to construct one from the other (as long as they compute in finite time), but they are tested in a different way: enumerative grammars are evaluated against their ability to enumerate all grammatical expressions, recognition grammars are tested by using datasets which contain both grammatical and ungrammatical expressions.[8] In this article we only examine enumeration grammars; see Brattico (2019, 2022) for an example of a minimalist computational recognition grammar that uses similar methodology.

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. Subcategorization, however, relies on a labeling. Let us begin with the following provisional head/labelling algorithm:

(2) *Label/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. recursive into Y.

---

[8] When applied to linguistic theorizing, however, enumerative and recognition grammars tend to have nontrivial differences. For example, 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, and in certain ways also misleading, but not unimportant when working with empirical linguistic theories.

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) 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) and indeed any other assumption as speculation; empirical justification will be based on a separate protocol, described below. The important point here, however, is that once we know the label/head for any constituent, 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. 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 (we discuss adjuncts in Section 5). Again, whether this definition of "specifier" is plausible will be determined by the subsequent testing and need not concern us here. Having introduced subcategorization, we must decide where it applies. The possibilities are: (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 option 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 examined if the derivations were broken down into separate sub-derivations, but since this option was not implemented, this alternative is irrelevant. Assumption (i) requires that we block

derivations violating subcategorization already inside the derivational search function. Thus, we add the subcategorization 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. As a consequence, derivational branches which lead into ungrammatical outcomes are blocked before they are created.

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 salad. Provided with reasonable subcategorization features 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 (mappings from derivations into output phrase structures into linearized sentences are all many-to-one).[9] The following screenshot shows what happened during the first three steps of the derivation:

[9] It is possible to filter the sentences on the basis of identical output, but the function is nontrivial due to the recursive step in the derivational search function which requires that all constituents that appear in the output solutions are different data objects (copies of the originals). The reason for this is that all derivational branches must remain independent derivations.

```
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 yet 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 an explicit set of *target sentences*, gold standard expressions that we want the grammar to enumerate to have a proper empirical justification. Moreover, both the numeration and the target sentences can be provided in 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 to run the simple VP-grammar against two target sentences:

```
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 shows that the it is observationally adequate. External files like this will become the *datasets* that we will use to 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 turns out to be observationally adequate, so 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. We could go one step further and assume that observational adequacy is tested only in this way, thus there will be no additional external justification.

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 $\rightarrow$ 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 that had only one operation Merge.[11]

[10] 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.

[11] This assumption is not empirically implausible and characterizes a whole range 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. Both are possible strategies.

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 $[_{YP}\ (_{Y}\ X\ Y)^{0}\ [...\sout{X}...]]$ 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 regular phrases and "copies" something from an existing structure. Let us add this operation, which we call head movement, into the theory.

First we stipulate a feature 'zero-level' to distinguish complex zero-level categories $(_{Y}\ X\ Y)^{0}$ from complex phrasal categories $[_{\alpha P}\ X\ Y]$ and modify the head algorithm such that it responds to this property and not only to the lack of daughter constituents. 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.[12] That is, regular Merge creates $[_{\alpha P}\ X\ Y]$, Head Merge $(X\ Y)^{0}$. This operation will be part of the final head movement operation. 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

[12] 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 allows us to encapsulate the code (=have it all in one place) that handles the creation of complex words. Moreover, instead of positing head movement, 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.

(2) so that it calculates heads also for complex zero-level categories and then let the features of complex heads $(X\ Y)^0$ 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 as a heuristic solution since the former requires a notion of adjunction that we will introduce only later. 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 produces new complex candidate words (e.g., *the#the*, *the#bite#man*) 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,[13] this does not match with the original specification $\alpha = [_{YP}\ (_{Y}\ X\ Y)^0\ [_{XP}\ ...\sout{X}...]]$ which copies X from within the merge partner XP. One possibility is that head movement is a grammatical operation like Merge and applies freely: α will become the composite operation of Merge + Head Move which apply occasionally in a sequence but do 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 syntactic objects in the syntactic working memory but also the daughters of the said elements.[14] 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 three issues that ultimately caused us to reject it for the purposes of this study. 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.

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

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

This forces us immediately to posit constraints to counterbalance the illicit derivations, so much so that the restrictions will effectively nullify the idea that head movement is a completely general operation. 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.[15] Finally, since our purpose is to examine the complexity properties of realistic linguistic theories, we should aim for test candidate grammars which minimize complexity. The above solution is extremely costly in terms of complexity.

A second solution is to posit a special category of syntactic rules into the derivational search function which targets single objects and tampers with their internal properties. Head movement, for example, would target some X, locate its head (or all heads) and apply 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 even implemented them for testing purposes. However, there is a third alternative: Head Move is part of Merge. We can assume that right before [Y XP] is created head movement is applied to X^0 (assuming the Head Movement Constraint (Koopman, 1984; Travis, 1984)) 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 $[_{YP}\ (X\ Y)^0\ [_{XP}\ ...\cancel{X}...]$. 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 satisfies HMC. We can perhaps think of head movement as some kind of "repair" operation

[15] 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.

executed before Merge. The operation does not increase the complexity of the grammar, since it applies whenever Merge applies and does not involve derivational branching.

To test this model against data let us posit a bound tense head T* into the lexicon and assume that it selects for a VP to create tensed verb phrases. To verify this grammar against data we assume the numeration {*T**, *the*, *dog*, *bark*} where *bark* is an intransitive verb and the target sentence is *the man bark#T** (dataset #3).[16] Running the grammar with these assumptions produces the following:

```
(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*'}
```

This experiment verifies 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 generated an VS word order that was not among the target sentences. The reason this grammar did not put the argument to the preverbal SpecTP position is because we assumed in the lexicon that intransitive verbs must have DP specifiers.[17] If we remove this assumption, the grammar generates the correct word order *the man barks* but with a questionable analysis:

```
(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 will most likely be considered implausible (in other words the grammar is observationally but not descriptively adequate). 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

[16] 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.

[17] Intransitive verbs can be divided at least into two categories, those which requires a DP-specifier ("unergatives") and those which require a DP-complement ("unaccusatives"). The choice does not matter for the matter at hand.

that the subcategorization features of verbal elements are checked inside the VP. We assume this model and return to the problem of English V-initial sentences below.

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. Consequently 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 direct consequence of the way head movement was defined. 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.

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. In fact, it follows more generally that $[(X\ Y)^0\ [...\cancel{X}...]]$ occurs if and only if Y selects for XP and is a bound morpheme.

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 two DP-arguments, it will pass SVO and VSO sentences with analyses (i) [$_{TP}$ T [$_{VP}$ DP [$_{VP}$ V DP]]] and (ii) [$_{TP}$ DP [T [$_{VP}$ DP V]] that one might and most likely will consider implausible. 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 base-generates one inside TP. Whether we want this to happen or not this shows that serious linguistic hypotheses must be *robust* in the sense that they pass several tests.[18]

English does not generate V-initial sentences, but some other languages do. To model language-specific crosslinguistic variation we need a theory of the said variation and some technical implementation for "different grammar." The easiest way to accomplish a system of this type is to create several language-specific speaker models instantiating the system 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 the variation 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.

[18] This is not an unimportant matter because while the entire force of the justification for our grammar or hypothesis will depend on the nature of the dataset(s), the selection of the relevant numerations still 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 the model is 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 in some manner.

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 to the specific position but is copied from within the complement to create $\alpha = [_{YP}$ ZP $[_{YP}$ Y $[_{XP}$ ~~ZP~~ [X WP]]]] in an operation we call *phrasal movement*. Implementing an operation of this type without countercyclic derivations requires that 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 in connection with head movement are possible as well and would seem 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 VS(O) sentences from the output while not adding anything to complexity. For example, the 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
```

First, T is combined with the verb to create a finite tensed verb; the resulting complex verb triggers an 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.[19] The model passes also transitive numerations but notice that these mechanisms do not yet correlate A-movement with any "anchoring" properties such as topicness, agreement or case 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.

[19] 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 formalized system.

In addition, not all phrasal movement is A-movement. Operator movement observed in English interrogatives such as *which man*$_1$ *did the dog bite* __$_1$ have different properties. Moreover, English interrogativization is associated with Aux-inversion, which is also visible in yes/no questions (*did the dog __ bite the man?*). Let us first fix the issue with the auxiliary. Suppose *did* represents T and that the force of the sentence (e.g., interrogative, declarative, imperative) is represented by an abstract C head subcategorizing for TP. We can test the results 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 morpheme, which generates the Aux-inversion pattern (dataset #10)[20]:

```
(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 are 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

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

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, 1982) follow in the sense that left phrases can never be searched. However, the numeration posited above does not quite 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 shows 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 generated a few ungrammatical sentences such as those in (4) which show that the model is not otherwise 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. Notice that if the interrogative pronoun was the subject, it was first moved to SpecTP and only then to SpecCP in a process called successive-cyclic movement (i.e. A-movement followed by A-bar movement). This consequence is unavoidable if we maintain that phrasal movement operates cyclically. 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 property of the grammar allows us to create a kernel model for English personal passives. If we assume that there exists a special "passive" v* with an unthematic specifier position (EPP) instead of an external thematic 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 [$_{v*P}$ *the man*$_1$ [$_{v*P}$ (*bite* v*) [$_{VP}$ ~~*bite*~~ __$_1$]]] (step 57) and which is then followed by A-movement to SpecTP (step 58).

In addition to Merge and Move, the standard minimalist theory posits a third operation Agree that is responsible for feature covariance dependencies, such as standard subject-verb agreement. This operation was not implemented. The reason was because the above system already contains a kernel of an operation of this type: we assumed that C(wh) searches for a matching 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, in some more recent models). 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 wanted to search for elements of specific type, such as grammatical subjects, a

feature-based search function would become an obvious candidate. It is possible that a theory of Agree could be developed from these mechanisms without positing a separate operation.[21]

5 Adjunction

In this section we consider the status of adjuncts and adjunction and formalize them as parallel objects residing in the syntactic working memory until linearization. We can also depict them as "virtual constituents" of their hosts.

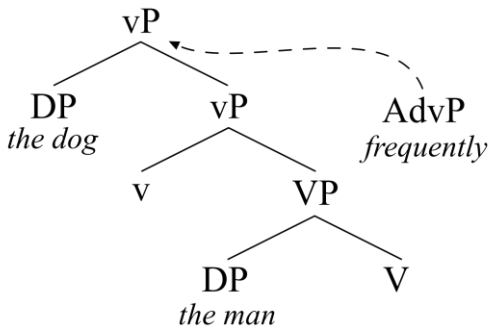
Consider a sentence such as *the dog bites the man frequently*. This construction behaves as a regular transitive clause but with an additional adverbial modifier *frequently* that is (i) invisible for all head calculations within the projectional spine of the transitive clause, (ii) optional and (iii) occurs in a curious rightward position that cannot be the complement position of the verb, being filled by the patient. Semantically the adverb (iv) modifies the whole event, not the man, the dog or even biting as such; what happens frequently is that the dog bites the man. Another property is that we can add more adverbials to the clause, seemingly without restrictions (v). If we assume that bare events are represented by VPs, then the adverb should be connected in some way to the verb phrase (vi), but there is nothing in our grammar that connects anything directly with phrasal level representations; all dependencies are formed between heads. In fact, there is nothing in the model that reproduces any of the properties (i–vi). In the minimalist theory as well as in many other grammars elements like the adverbial *frequently* are treated as *adjuncts*, elements that are merged inside grammatical projections in a more loose manner.[22] Let us add this feature into the grammar.

---

[21] This approach would 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, so we did not develop this hypothesis further.

[22] Some grammars collapse specifiers and adjuncts into the same category, but we provide a sperate adjunct category in order to develop a fully general model. In a grammar without separate category of adjuncts the option can simply be removed or alternative the notion of specifier could be based on adjunction.

So far we have assumed that the derivational search function takes the contents of syntactic working memory as input, applies an operation OP to a pair of syntactic objects X and Y and replaces them with the outcome OP(X, Y). The syntactic working memory therefore contains a set of "parallel" syntactic objects that we can think of being under active consideration at any given movement during the derivation. We can imagine, however, that some operations leave the results into the sWM as parallel objects, which then become "adjuncts." Let us assume that there is an operation Adjoin(X, Y) which connects X to Y via one-way mother-of dependency so that Y will be the mother of X but leaves the two elements otherwise intact in the sWM. Thus, the mother of *frequently* could be a verb phrase while all other properties of the verb phrase, including its daughter constituents, remain the same:



Let us assume that when X is connected in this way to Y by adjunction, the result is notated as Y|X. For example, the adverb *frequently* would be notated as vP|*frequently* when it is linked with the verb phrase as its host structure. Crucially, we allow X to live inside sWM as a parallel structure and do not embed it fully inside Y.

Not all syntactic objects can be adjoined, and those which can be adjoined can and often do have a limited distribution. For example, we want to create a theory in which VP-adverbs such as *frequently* must be adjoined inside the VP but not inside PP. To both restrict the class of adjoinable objects and their distributions let us assume that adjoinability is defined by a feature α:F where

feature F must occur inside the head of the projection to which the adjunct is linked with.[23] A VP-adverb would therefore be defined by feature α:V. Only syntactic objects headed by this feature can be adjoined. We do not assume the he vice versa: adjoinable objects can also be merged, but must then satisfy subcategorization.[24] Putting some of the additional technical details aside, let us consider a simple intransitive sentence *the dog barks frequently* with the numeration {T, *the*, *dog*, *bark*, *frequently*} and where we assume that the adverbial has feature α:V. The output (dataset #17) is as follows:

```
(1) the dog bark#T frequently  [_TP [_DP the dog ]:2  [_TP (bark T ) [_VP __:2  [_VP __  frequently ] ] ] ]
(2) frequently the dog bark#T  [_AdvP frequently [_TP [_DP the dog ]:1  [_TP (bark T ) [_VP __:1  __  ] ] ] ]
(3) the dog bark#T frequently  [_TP [_DP the dog ]:1  [_TP (bark T ) [_VP __:1  __  ]* ] ] + { VP|frequently° }
(4) the dog bark#T frequently  [_TP [_DP the dog ]:1  [_TP (bark* T ) [_VP __:1  __  ] ] ] + { V|frequently° }
(5) the dog bark#T frequently  [_TP [_DP the dog ]:2  [_TP (bark T ) [_VP __:2  [_VP __  frequently ] ] ] ]
(6) the dog bark#T frequently  [_TP [_DP the dog ]:1  [_TP (bark* T ) [_VP __:1  __  ] ] ] + { V|frequently° }
Derivational steps: 106
Errors 0
```

Parallel adjuncts are represented as + {X, Y, ...} after the root structure. Solutions (1, 2–6) are generated by either merging the adverbial to the complement position of the verb (1, 5) or by adjoining it either to the VP (3) or V (4, 6). Solution (2) is generated by merging the adverb above TP, but although this sentence is grammatical the analysis is not plausible since the head algorithm will regard the whole phrase as an adverb phrase.[25] Notice how the adverb is written outside of the host structure as a parallel object, with its link with the host structure now expressed by the X|Y notation (the host is marked by an asterisk). That is, *frequently* is not an ordinary constituent of the VP but is connected to it virtually only by a mother-of dependency. All label/head calculations inside the host structure remain the same, while the adjunct is completely invisible. The newly introduced operation Adjoin is visible in the log file:

[23] A possible generalization is α:F_1, . . ., F_n in which several features must be checked.

[24] This property can easily be removed from the grammar, but is arguable needed to model properties of DPs and PPs which can assume both roles.

[25] This consequence can be avoided if we assume that *frequently* is derived as [$_{AdvP}$ (*frequent-ly*)^0[$_{AP}$ ~~*frequent*~~]], but we do not go into the details here.

```
53.

    sWM: frequently°, [_VP [_DP the dog ] bark ], T°
    Adjoin(frequently, [_VP [_DP the dog ] bark ])
    sWM′: [_VP [_DP the dog ] bark ]*, T° + { VP|frequently° }

54.

    sWM: [_VP [_DP the dog ] bark ]*, T° + { VP|frequently° }
    Merge(T, [_VP [_DP the dog ] bark ]*)   Head Chain (T, bark)
    Chain ((bark T ), __:1 )

    sWM′: [_TP [_DP the dog ]:1  [_TP (bark T ) [_VP __:1  __  ]* ] ] + { VP|frequently° }

    [_TP [_DP the dog ]:1  [_TP (bark T ) [_VP __:1  __  ]* ] ] + { VP|frequently° }
    ^ ACCEPTED: the dog bark#T frequently
```

The VP-adjunction is performed in the step 53, after which the addition of T (step 54) results in a grammatical sentence. Running the model with a transitive numeration shows, however, that the adverb is always positioned between the verb and the object (*the dog bite frequently the man*), while it should be positioned after the object. This brings up the question of linearization. In the above simulations we assumed (thus far without explicit comment) that the adjunct is always linearized to the left of its host, before any other constituent. It appeared at the post-verbal position in the intransitive clause because we nowhere assumed that verb copying should update the virtual adjunct link; thus, the adjunct is still linked to the silenced verb at the base position and is linearized in relation to that element as shown in (5).

(5) The dog (bark$_1$, T) [$_{VP}$ frequently __$_1$].

The result is the same if the adjunct is linked with the verb phrase. Changing the linearization to right will handle both intransitive and transitive clauses, but will create problems later because it makes all adjectives (which we assume to be adjuncts, not uncontroversially) postnominal. More generally, we may want to have both left and right adjuncts. Since the linearization of adjuncts is subject to language-specific variation, we can perhaps assume that it is controlled by a lexical linearization feature. If we assume that adverbs like *frequently* are linearized to the right, the grammar will generate the variants in (6) after ~80k derivational steps (dataset #18).

(6) a. The dog bites the man frequently. (Adverb adjoined to VP)

b. The dog bites frequently the man. (Adverb adjoined to V)

c. Frequently the dog bits the man. (Adverb merged to TP)

If adjectives are linearized to left, they will occupy a position between D and N. A numeration {*the*, *angry*, *dog*, *bark*, T} will generate the following output (datasets #18, 19) if we restrict adjective distribution to NPs:

```
(1) the angry dog bark#T  [_TP [_DP the dog* ]:1  [_TP (bark T ) [_VP __:1  __  ] ] ] + { N|angry° }
Derivational steps: 55
Errors 0
```

The derivation for *the angry dog* is as follows:

```
4.

    sWM: the°, angry°, dog°
    Adjoin(angry, dog)
    sWM′: dog*°, the° + { N|angry° }

5.

    sWM: dog*°, the° + { N|angry° }
    Merge(the, dog*)
    sWM′: [_DP the dog* ] + { N|angry° }

    [_DP the dog* ] + { N|angry° }
    ^ ACCEPTED: the angry dog
```

First *angry* is adjoined to *dog* (step 4), then the definition article is merged with N to generate [$_{DP}$ D N] with the adjective existing as a parallel object linked with N (step 5). Linearization to left creates the final output sentence. We could create postnominal adjectives by assuming that they linearize to the right of their host.

Implementing the above adjunct analysis involves several technical questions that only concern the implementation. For example, linearization is much simpler to implement and easier to understand if the host structures have pointers to their adjuncts, so they were added as an additional property to the syntactic objects. These pointers are not part of the linguistic theory or the (intended) algorithm, rather they belong to the implementation layer. It is assumed that in an empirically more realistic theory adjuncts would be “collapsed” to ordinary constituents before linearization. Currently the system is implemented by allowing linearization algorithm to access the

adjunct pointers so this extra step, which is nontrivial, does not have to be implemented. An additional reason to remain more cautious is because essentially the same adjunction mechanism could be implemented by marking constituents specifically as being adjoined and then ignore those constituents in the head calculations.

Having adjunction in the model opens up several new derivations, for example, we can provide PPs an ability to right-adjoin to verb phrases. A vP-numeration such as {*the, dog, v, bite, the, man, in, the, city*} generates 13608 verb phrase configurations after 4,5 million calculation steps if we allow the PP to right-adjoin to any verb (phrase); an intransitive variation is much easier to calculate and was thus included into the datasets (#20). It produces the expected output, part of which is shown below:

```
(10) the dog bark in the city  [_VP [_DP the dog ] bark ]* + { VP|[_PP in [_DP the city ] ] }
(11) the dog bark in the city  [_VP [_DP the dog ] [_VP bark [_PP in [_DP the city ] ] ] ]
(12) the dog bark in the city  [_VP [_DP the dog ] bark* ] + { V|[_PP in [_DP the city ] ] }
```

Solution (11) is generated by merging the PP to the complement of V, (10) adjoins the PP to the VP, (12) to the verb. Adjunction to the verb could be prohibited by further stipulation or feature, but was left into the model to maintain full generality. Linearization was set to right for all English prepositions; notice that leftward PPs are still possible via base-generation to the specifier position whenever the operation is licensed by subcategorization. Replacing *the* = *which* demonstrates that the other half of the CED-effects also follows: there are no derivations in which the interrogative is extracted from an adjoined PP: only subjects and objects can be fronted to SpecCP. However, the model does not handle pied-piping and therefore strands the preposition (7)(#dataset 21).

(7) ?*Which city$_1$ did the dog bark [$_{PP}$ in __$_1$]].

Notice that the correct sentence *in which city did the dog bark?* is also in the output set, but it is generated by base-generating the interrogative PP into the SpecCP and could be considered implausible. How to capture pied-piping and prevent stranding is a separate problem; either there is a wh-feature at the P due to “percolation” or due to some other operation that takes place when P is

merged with the DP or copying targets a larger category than the projection from the head containing the wh-feature. There is a large literature addressing this matter that we cannot attempt to review here.

6 Summary

Table 1 summarizes the amount of calculations required to verify the minimalist test grammar constructed here as a function of the size of the numeration or data complexity. We provide the numbers both for realistic words and empty words, the latter which represents the worst case scenario. Notice that both head and phrasal movement were implemented as part of Merge and did not increase complexity. Moreover, the numbers provided for the realistic simulation concern the particular grammar formulated in this article, will consequently change depending on the grammar and the numeration, and can only be determined by computational experimentation; however, since the slope of the complexity function is exponential, these examples are still able to provide a good estimation of what to expect.

Table 1. Number of calculation steps as a function of the size of the numeration and data complexity.

		Number of steps	
Size	Example	Realistic	Worst case
1	*the*	0	0
2	*the man*	1	2
3	*the dog barks*	2	18
4	*the dog barks#T*	10	228
5	*C the dog barks#T*	96	4580
6	*C the angry dog barks#T*	680	137430

7	*the dog bite#T#v the man*	3948	5772102
8	*C the dog bite#T#v the man*	37076	hundreds of millions
9	*C the angry dog bite#T#v the man*	1089047	billions
10	*the dog bite#v#T the man in the city*	34256787	

All paper-and-pencil methods quickly become infeasible after the simplest sentences, but notice that eventually even fast computers will hit the wall. The Python script used here calculated at the rate of 10 steps/millisecond = million steps/two minutes when we removed all printing and logging. One practical solution is to restrict the admissible operations manually at the stage when the numeration is constructed, especially when testing hypotheses provisionally, but this of course carries a risk that some illicit behavior remain latent. The phase theory of Chomsky (2000, 2001) is a variation of the same idea and provides another avenue. In this model the derivation of expressions is divided into sub-numerations ("phases") each derived independently and then tested for combinations in a separate step. Thus, we restrict in advance what elements can be combined with which elements. Since all outputs of all sub-numerations must be combined with each other, it is not a priori clear how much this method can flatten the complexity function, and an additional concern is that the theory makes empirical predictions that must be justified against datasets. We leave this question for a future study.

7 Conclusions

This article examined the complexity problem of what we consider representing a fairly conservative or standard minimalist grammar and proposed a computational solution. Even with a tightly constrained and (in our view) realistic grammar the number of calculations required to show that the theory matches with the data is too high to be accomplished with any paper-and-pencil methodology, leaving us with no choice but to use mechanical methods such as the one proposed here or some other conceivable alternative. Notice that in a realistic linguistic project it is rarely

sufficient to show that the grammar matches with a few target sentences: the testing must be repeated after any change in the lexicon/theory and executed for several numerations and target sentences.[26] This raises the requirements to the level of hundreds of millions, if not more. The only alternative is to opt out altogether from testing grammars and linguistic hypotheses in a rigorous way, but this is clearly a completely unwarranted research strategy that no respectable scientific field would accept.

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

[26] While batch testing is easy to implement technically by writing the tests into the one external input file, as was done here, it 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.

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.

Brattico, P. (2019). *Computational implementation of a linear phase parser. Framework and technical documentation (version 15.0)*. Pavia.

Brattico, P. (2022). Predicate clefting and long head movement in Finnish. *Linguistic Inquiry*, *54*(4), 663–692. https://doi.org/http://dx.doi.org/10.1162/ling_a_00431

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.

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.

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. (1982). 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.

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.