Towards a discovery procedure for minimalist grammars[*]

Marina Ermolaeva

Lomonosov Moscow State University

mail@mermolaeva.com

October 2022

Abstract

In this paper I examine whether solutions to syntax puzzles found in theoretical literature can also be derived in a principled way and on quantitative grounds. First, I outline an algorithm for refining "naive" descriptions of linguistic data, expressed as Stabler's (1997, 2001) minimalist grammars, with respect to a quantitative evaluation measure. The proposed procedure is built around the idea of lexical item decomposition and focuses on identifying and factoring out phonological and syntactic redundancies in a grammar. To demonstrate this in action, I report the results of an experiment with a prototype implementation of the algorithm, using l-selection of prepositional phrases in English as a case study.

Keywords: syntax, morphology, discovery procedure, evaluation measure, minimalist grammars, lexical item decomposition

1 Introduction

For Chomsky (1957), for every natural language there exists a single correct grammar, and the goal of a linguist is to find this unique grammar based on a given data set (corpus of utterances) drawn from the language in question. As the strongest requirement that could be placed on linguistic theory, he defines the *discovery procedure* as a "practical and mechanical" way to obtain the correct grammar for a given corpus of data. The search for the discovery procedure is initially dismissed as "unreasonable"; instead, Chomsky proposes the "more modest goal" of developing an *evaluation procedure*, a method of determining which of

[*]This paper is based on parts of the author's PhD dissertation (Ermolaeva 2021a).

two given grammars is better with respect to a specific data corpus. However, a similar idea appears in (Chomsky 1965), where an acquisition model (in the context of how a child learns a language) is defined as follows:

i. a universal phonetic theory that defines the notion "possible sentence"

ii. a definition of "structural description"

iii. a definition of "generative grammar"

iv. a method for determining the structural description of a sentence, given a grammar

v. a way of evaluating alternative proposed grammars

(Chomsky 1965, p. 31)

In other words, we need to characterize possible grammars in a precise way, specifying how they assign structural descriptions to sentences (i–iv). The last requirement (v) includes an evaluation procedure as well as some realistic way for it to select between possible grammars.

This paper aims to take a step towards constructing an automated ("practical and mechanical") discovery procedure for natural language syntax, inspired by the above definition. One task of particular interest is to examine which solutions to syntax puzzles found in current theoretical literature can also be derived in a principled way and on quantitative grounds. To this end, the definition of possible grammars must be compatible with the current iteration of generative syntax – the Minimalist Program of (Chomsky 1995, 2000). Minimalist grammars or MGs (Stabler 1997), designed as a formalization of Minimalist syntax, are a natural choice of a formalism to encode syntactic analyses.

What will serve as input for such a procedure? One approach, well-represented in the literature, is to start with a corpus of linguistic data – presented as unlabeled and unstructured strings (Clark and Eyraud 2007; Yoshinaka 2011; Clark 2017) or annotated with additional information such as semantic roles and agreement relations (Indurkhya 2019). Using de la Higuera's (2010) terminology, such tasks can be referred to as *grammar induction* (when the primary goal is to obtain a grammar that explains the data) or *grammatical inference* (when a true target grammar is expected to exist, and the focus is on the quality of the learning process itself).

In order to focus on distinctions between different possible analyses of the same phenomena, I adopt a different setting. Rather than a corpus, the starting point is a grammar, given in the same format as expected output but defined in a maximally theory-neutral way. In particular, the input grammar postulates no empty functional heads and no structure within words. The goal of the procedure is to refine the grammar, derive abstract elements as needed and produce a linguistically plausible description accounting for the original data. This shifts the focus to capturing generalizations within an already present grammar. To distinguish this task from grammar induction or grammatical inference, I will use the term *grammar optimization*.

The paper is structured as follows. Section 2 briefly describes minimalist grammars (Stabler 1997, 2001) and discusses the similarities and differences between them and the machinery assumed in theoretical proposals. Section 3 introduces operations over minimalist grammars as the driving force behind syntactic analysis and formalizes the notion of a linguistic generalization in the domain of syntax. It then provides a short description of an algorithm for grammar optimization built around these operations. Section 4 presents the results of an experiment, using l-selection of prepositional phrases in English as a case study. Finally, Section 5 provides some discussion and perspectives for future work.

2 Minimalist grammars

2.1 Lexical items, Merge, and Move

Consistent with the increased role of the lexicon in Minimalist syntax, a minimalist grammar is defined by its *lexical items*, or LIs. Each lexical item consists of a *string component* (of phonological segments, often approximated by orthography) and a *feature bundle* – a sequence of syntactic features encoding its selectional properties. Lexical items are considered simple (atomic) syntactic expressions. Syntax is driven by feature matching and checking; complex expressions are built via two structure-building operations, Merge and Move.

A feature of the form `x` corresponds to a syntactic category, whereas `=x` and `x=` are selecting features which indicate that an expression is looking to Merge (on the right or on the left, respectively) with one of that category. An expression carrying the feature `-x` is required to Move at some point in the derivation, and `+x` attracts a sub-expression whose head carries that feature into its specifier position; `*x` triggers covert Move, which acts in a similar way but leaves the string component of the moving sub-expression in the original position. Features of the form `=x`, `x=`, `+x`, and `*x` together are called *attractors*; `x` and `-x` are *attractees*. Matched features are *checked* (deleted) and no longer visible to syntax; we will keep them in representations for clarity, marked as $\boxed{\texttt{x}}$. The next feature in each sequence, if present, becomes available for syntactic operations.

An example grammar of five lexical items is given in Figure 1.

this	::	`=n d -k`
boy	::	`n`
is	::	`=g +k t`
laughing	::	`=d g`
laughs	::	`=d +k t`

Figure 1: A toy MG

Let us consider two LIs from this grammar: *this* :: `=n d -k` and *boy* :: `n`. Their feature bundles start with the matching features `=n` and `n`, respectively. Merging these two LIs produces the complex expression shown in Figure 2. Whichever expression contributed the attractor (in this case, *this*) becomes the *head* of the newly formed expression. This is recorded by the label of the parent node: $<$ if the head is on the left, $>$ otherwise.

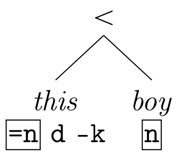


Figure 2: Merge(*this*, *boy*)

The head of this complex expression carries a feature bundle whose first unchecked feature is `d`. Merging it with the LI *laughing* :: `=d g` results in the expression in Figure 3 (and checks `=d` on *laughing* and `d` on *boy*). The new expression, in turn, is compatible with *is* :: `=g +k t`; this Merge step produces Figure 4.

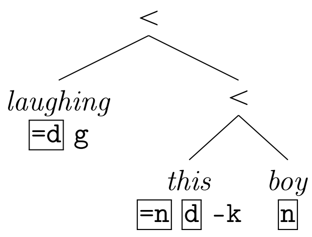


Figure 3: Merge(*laughing*, 2)

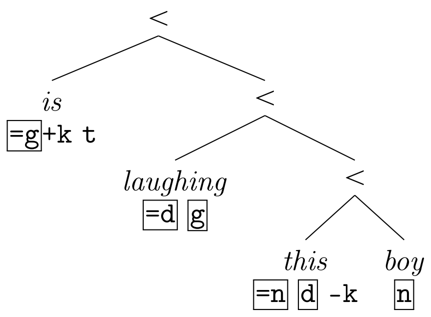


Figure 4: Merge(*is*, 3)

Unlike Merge, which combines two expressions, Move is unary. In Figure 4, the matching features are `+k` on *is* and `-k` on *this*. The sub-expression headed by *this* is attracted into the specifier position of *is*, which becomes the head of the new expression, Figure 5.

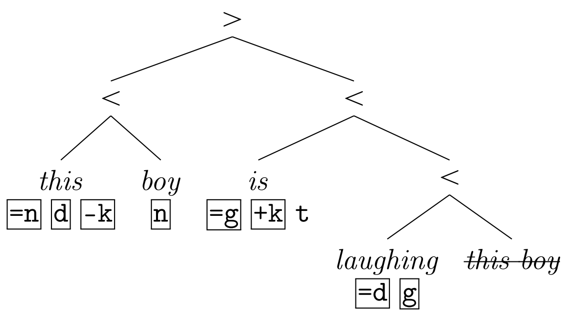


Figure 5: Move(4)

All syntactic features within this expression have been checked, except the category `t` on its head. We will call this a *complete* expression of category `t`.

This toy grammar generates only two sentences: *this boy is laughing*, which is the string yield of Figure 5; and *this boy laughs*, which can be produced by having *laughs* take Figure 3 as its complement. For larger MGs with dozens or hundreds of LIs (and potentially generating an infinite set of sentences) it is often convenient to visualize possible head-complement relations as a directed multigraph whose vertices correspond to category features, and edges to lexical items. Figure 6 gives the multigraph representation of Figure 1.

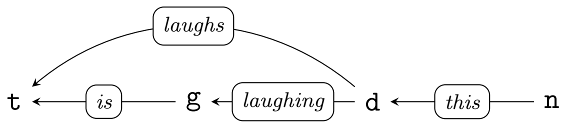


Figure 6: Head-complement relations within Figure 1

This graph does not reflect all relations between LIs in the grammar, since it ignores Move and any specifiers formed by Merge. Thus, LIs without any selecting features (such as *boy* :: `n`) don't contribute an edge to the graph. Each path from `n` to `t` indicates a possible sequence of LIs along the clausal spine. Multiple paths between vertices indicate that there is more than one option available at that point in the derivation. For instance, there is an edge connecting `v` and `t`, as well as an alternative path between these categories. This reflects the fact that a DP (expression of category `d`) can be selected either by *laughs* :: `=>d +k t` or by *laughing* :: `=>d g`, in the latter case producing a valid complement for *is* :: `=g +k t`.

At a glance, MGs appear rather different from mainstream Minimalist syntax in numerous ways – with respect to the feature calculus, implementation of movement, locality, and other issues. However, many of these seeming points of disagreement are a matter of convenience and can be tweaked without altering the crucial computational properties of the formalism, and much of the machinery employed by Minimalist syntax can be translated into the bare-bones MGs or implemented as extensions. For an in-depth discussion of how faithful MGs are to Minimalist syntax, see (Graf 2013, pp. 96–125). Here, I will briefly focus on one example: the hierarchy of syntactic categories.

In order to bridge the apparent gap, let us examine an explicit theory of feature structures compatible with Chomsky's framework, proposed by Adger (2010). Lexical items are defined as sets of category features (T, V, N, D, ...) and morphosyntactic features (case, number, person, ...), each specified as bearing a value (drawn from some finite set) or being unvalued. Category features are ordered according to a number of universal Hierarchies of Projection (HoPs). The definition of a well-formed syntactic object specifies that the projecting head must bear a higher-valued category feature than the dependent; HoPs thus constrain the building of structure.

As an illustration, consider three lexical items: *the*, *many*, *men*. In Adger's system, there is a HoP in the nominal domain specifying N < Num < D. By assigning the categories D, Num, and N to lexical items *the*, *many*, and *men* respectively, the system ensures that *the many men* and *the men* are well-formed, while **many the men* is ruled out.

How would such a constraint translate into MGs? One option is to extend the formalism to include an explicit partial order relation over *Base*, the set of categories. However, an implicit ordering is already present in the bare-bones MGs. Consider the LIs in Figure 7.

the :: `=num d -k`
many :: `=n num`
ϵ :: `=n num`
men :: `n`

Figure 7: An MG ensuring the correct ordering of nominal projections

Since Merge is feature-driven, the ordering of lexical items within expressions generated by this grammar is already constrained by their feature bundles: *the* can select an expression headed by *many*, but not vice versa. Finally, all we need to ensure the optionality of *many* is a phonologically empty lexical item, ϵ :: `=n num`. It represents the idea that expressions of category `num` have a more limited distribution than those of category `n`; that is, any expression that selects a `num` can also select an `n`, but not the other way around.

Having no explicit machinery to encode HoPs significantly increases the size of a grammar. This is an example of a trade-off: the conceptual simplicity of bare-bones MGs is preserved at the cost of a less elegant solution to a specific problem, with the understanding that the formalism can be extended, if necessary, to accommodate this additional machinery,[1] and that doing so would not change its core computational properties.

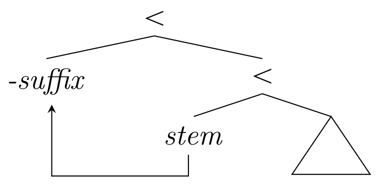
2.2 Complex words

The basic formalism outlined so far does not offer a way of recognizing structure within words, and the grammar in Figure 1 treats each word as an inseparable unit. This is an oversimplification, which leads to considerable redundancy and missed generalizations. For instance, the grammar does not reflect the fact that *laughing* and *laughs* have a common root, or that *laughs* shares certain syntactic properties with *is*.

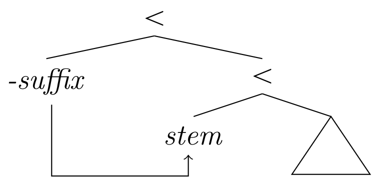
To remedy this, we need a way to combine multiple syntactic heads into complex units corresponding to multimorphemic words. This gives rise to a number of questions regarding the syntax-morphology interface and morphology proper:

1. Where in the sentence are the complex syntactic heads pronounced?
2. How is morphological information transmitted between words?
3. How are the complex heads mapped to strings?

With respect to (1), multiple options have been explored in the literature. *Head movement* creates a chain of heads that is pronounced in the highest head position (Figure 8a). *Lowering* or *affix hopping*, on the other hand, allows an affix to attach to the head of its complement, with the whole word being pronounced in the lower position (Figure 8b).



(a) Head movement



(b) Lowering

Figure 8: Building complex heads

Both processes are widely taken to be present in English. In particular, finite auxiliaries undergo head movement to T, while finite lexical verbs have affixes lowered onto them. Evidence of this is presented by Pollock (1989): finite auxiliaries precede sentential negation and undergo subject-auxiliary inversion in questions, whereas lexical verbs do not.

[1]An extension adding a hierarchy of projections to MGs is implemented in (Fowlie 2013), which augments them with a partial order over selectors to handle adjunction in a concise and explicit way.

Unification of head movement and lowering is one of the defining features of Brody's (2000) Mirror Theory. This operation creates morphological dependencies between heads (lexical items), combining them into a *morphological word*. Each head is characterized as *weak* or *strong*. A complex morphological word is pronounced in the position of its highest strong head, or the lowest head if it does not contain any strong head. In a similar vein, Arregi and Pietraszko (2018) propose a generalized account of head movement and lowering as high and low spellouts of a single syntactic operation, *unified head movement*, supporting it with evidence of successive cyclic lowering as well as lowering feeding head movement. This proposal differs from Mirror Theory in that the default position of a complex word, in the absence of strong heads, is the *highest* head position. Among other things, this immediately offers an analysis for the English data. Tense, auxiliary *be*, and auxiliary *have* are weak heads, whereas lexical verbs are strong. As a result, finite *have* and *be* are pronounced in the highest position, since they don't contain any strong heads, but finite lexical verbs are pronounced in the lowest position.

Some approaches to complex heads have been adapted for MGs. Stabler (2001) incorporates both head movement and lowering into MGs as subtypes of selector features. This formalization renders the material in a lowered complex head inaccessible for any future head movement or lowering. Brody's framework was adapted into minimalist grammars by Kobele (2002), and was proven not to affect the weak generative capacity of the formalism. Arregi and Pietraszko's (2018) proposal is similarly implemented in (Kobele to appear).

Question (2) deals with local and long-distance flow of information between words and its morphological manifestation. Chomsky's Minimalist program considers these phenomena an effect of a general mechanism called Agree, which is driven by feature matching and forms relations between lexical items. The standard version of Agree takes place between a probe and a goal with matching features, such that the probe c-commands the goal and there is no other eligible goal that is closer to the probe. To establish an Agree relation, the probe look downwards into its domain (sister), and the goal transmits feature values upwards to the probe. In Adger's (2010) system, this is made explicit as a standalone operation which targets morphosyntactic features and establishes dependencies within existing structure – in contrast to Merge and Move, which operate on category features and build new structure.

A framework for agreement compatible with MGs is outlined in (Ermolaeva 2018; Ermolaeva and Kobele 2022). This line of work utilizes a more refined definition of lexical items: instead of a string component, each LI is associated with a set of morphological features and values, which encode information needed to realize the head as a string. Morphological features are then valued in the course of the derivation, using dependencies formed by Merge and Move. For instance, rather than the string *boy*, the corresponding LI would be represented by a set containing, among others, the features `number`:*singular*, `person`:*3*, and `case`:*unvalued*, whereas the lexical item *be* would carry `number`:*unvalued*, `person`:*unvalued*, and `case`:*nominative*. Their respective `-k` and `+k` features would be marked as allowing transmission of feature values along the Move dependency. Like many other modifications of MGs, operating over sets of morphological features provides a succinct way of formulating generalizations but does not change the core properties of the formalism.[2] More generally, as shown by Graf (2013), restrictions on LI compatibility can be added to MGs as long as they are definable in monadic second-order logic. In this case, the constraints can be built into standard MGs by refining syntactic categories. These results are used directly, for instance, in (Laszakovits 2018) to implement dependent case in MGs.

Finally, (3) falls outside the domain of syntax, or even the syntax-morphology interface, and into morphology proper. While this issue is largely outside the scope of this paper, it represents another decision which must be made to ensure that the syntax formalism we are using is capable of supporting complex words. Generally speaking, this requires an MG-compatible theory of morphology, understood as a function mapping words (simple or complex heads output by an MG) to phonological strings. This function may be as simple as string concatenation or as involved as a faithful formalization of a morphological framework proposed in the literature.

The various options available with respect to each of the three questions are meaningful and worth exploring in the context of grammar optimization. That said, just as with syntax proper, one needs to strike a balance between faithfulness and conceptual simplicity when making additions to the formalism. For now I make the following simplifying assumptions:

- All complex heads are formed by head movement;
- Lexical items carry string components rather than sets of morphological features;
- Morphology is concatenative, and all affixation is suffixation.

Following Stabler (2001), we introduce an additional subtype of selecting features, `=>x`, which triggers right Merge accompanied by head movement – which is implemented as concatenation of the heads of ghe two expressions. We will refer to lexical items bearing these selector features as *affixes* and write their string components starting with a hyphen.

[2]For example, subject-verb agreement in English can be expressed, in a straightforward but cumbersome fashion, as a bare-bones MGs over immutable strings:

I :: d -k$_{1SG}$	*boy* :: n$_{3SG}$	*am* :: =g +k$_{1SG}$ t
this :: =n$_{3SG}$ d -k$_{3SG}$	*boys* :: n$_{3PL}$	*is* :: =g +k$_{3SG}$ t
these :: =n$_{3PL}$ d -k$_{3PL}$	*walking* :: =d g	*are* :: =g +k$_{3PL}$ t

Here only lexical items with compatible string components may combine via Merge or Move. This is made possible by introducing a separate syntactic feature for each configuration of morphological properties resulting in a distinct morphological form.

With this addition, we can express the generalization that *laughs* and *laughing* share a root. To achieve this, we replace these lexical items with three new ones:

- the root *laugh* :: `=d v`, where `v` is a *fresh* (previously unused) category feature;
- two affixes that can select it: *-ing* :: `=>v g` and *-s* :: `=>v +k t`.

The modified grammar is shown in the multigraph form in Figure 9.

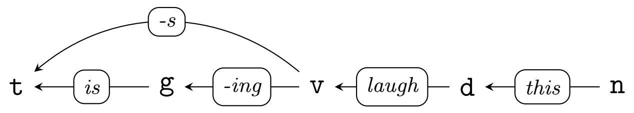
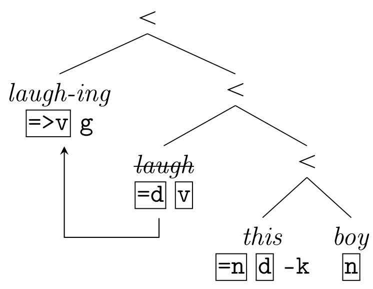
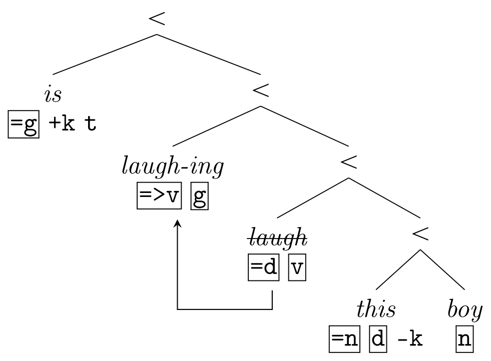


Figure 9: Head-complement relations: replacing *laughs* and *laughing* with *laugh*, *-ing*, and *-s*

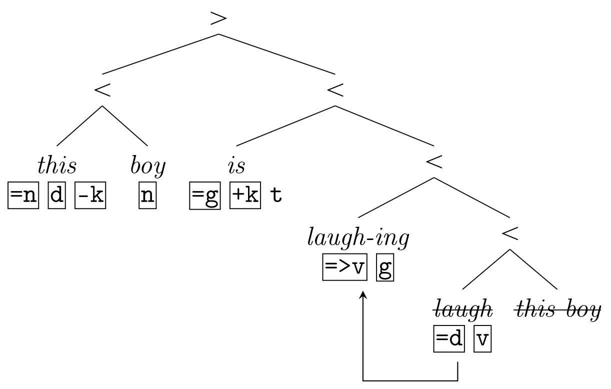
In order to derive the sentence *this boy is laugh-ing* with the new grammar, we construct Figure 3 as before. This expression is then merged with *-ing*, producing Figure 10a. Its selector feature, `=>v`, triggers head movement, concatenating *laugh* and *-ing* together (Figure 10a). The two remaining steps, Merge and Move, proceed as normal, resulting in the complete expression in Figure 10c.



(a) Merge(*-ing*, 3)



(b) Merge(*is*, 10a)



(c) Move(10b)

Figure 10: Derivation of *this boy is laugh-ing*

3 Optimizing grammars[3]

3.1 Decomposition and simplification

Let us consider more carefully how adding affixes to the formalism enables us to express generalizations in the grammar. In the previous section, we started with a grammar over whole words. In particular, it analyzed *laughing* :: `=d g` as a single LI, which took a DP as its complement and produces an expression of category `g` (Figure 11).

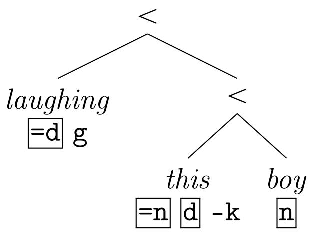


Figure 11: Single-LI *laughing* (=Figure 3)

With this addition, we can split this LI into a lexical verb *laugh* :: `=d v` and a suffix *-ing* :: `=>v g`. The former is responsible for merging in the argument DP, whereas the latter selects the lexical verb and produces an expression of category `g` (Figure 12).

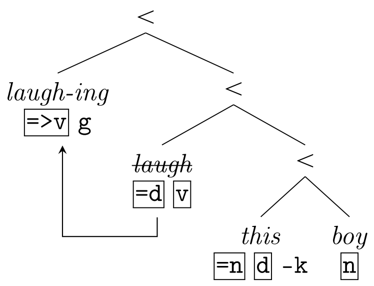


Figure 12: Decomposition of *laughing* into *laugh* and *-ing* (=Figure 10a)

A general assumption in Distributed Morphology (adopted, for instance, in (Marantz 1997; Embick and Marantz 2008)) is the idea that roots are category-neutral and must merge with a category-defining functional head, so every word is at least bimorphemic. This is easy to translate into MG lexical items. Continuing with the same example, we can further split the lexical verb *laugh* by assigning its entire phonological component to one LI, and its syntactic features to another. This move produces a root *laugh* :: `r` and a categorizing head -ϵ :: `=>r =d v` (where ϵ denotes the empty string). Once again, the information carried by the original

[3]A version of this section, along with some additional examples, is presented in (Ermolaeva 2021b).

LI is divided between the two new ones: the acategorial root contains the phonological content, while the silent categorizing head supplies the syntactic features (Figure 13).

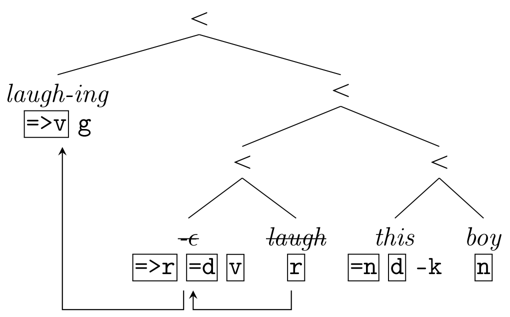


Figure 13: Decomposition of *laugh* into *laugh* and -ϵ

All three expressions are equivalent for syntactic operations, and can be used to generate the same sentences. However, they postulate a different amount of structure within words.

Lexical item decomposition, proposed by (Kobele to appear), is a generalization of this idea. In short, an arbitrary minimalist LI can be replaced with a stem and an affix by splitting both its syntactic feature bundle and phonological component. The stem carries a previously unused category feature, while the affix is given an attractor feature selecting for that category and triggering head movement.

In order to make a generalization, decomposition needs to process a set of lexical items simultaneously and factor out similarities between them. We will refer to this strategy as *batch decomposition*. An example is given in Figure 14. It starts with three lexical items, splits off the elements that they have in common – the prefix *laugh* and the syntactic feature `=d` – and expresses them as a new lexical item, *laugh* :: `=d x`, which is reused to construct every word in the batch. This batch decomposition step encodes the idea that *laugh*, *laughing*, and *laughs* share a stem – at the cost of postulating a single new syntactic category, `x`. This is an example of *left decomposition*, as the shared elements are on the left-hand side.

laugh :: `=d v`			-ϵ :: `=>x v`
laughing :: `=d g`	$\rightsquigarrow$	*laugh* :: `=d x`	*-ing* :: `=>x g`
laughs :: `=d +k t`			*-s* :: `=>x +k t`

Figure 14: Left decomposition

Similarly, shared affixes can be obtained via *right decomposition*, by factoring out common elements on the right-hand side (Figure 15).

dances :: `=d +k t`		*dance* :: `=d x`		
laughs :: `=d +k t`	$\rightsquigarrow$	*laugh* :: `=d x`		-*s* :: `=>x +k t`
jumps :: `=d +k t`		*jump* :: `=d x`		

Figure 15: Right decomposition

In many cases, batch decomposition alone cannot completely eliminate redundancies in a grammar. This is illustrated by the grammar in Figure 16. Note that we follow the simplifying assumptions made in Subsection 2.2 and abstract away from morphological irregularities of English and replace each word with a sequence of morphemes formed by string concatenation; *is* is rendered as *be-s*, and *will* as *will-s*.

Mary :: `d -k`
-*s* :: `=>m +k t`
-ϵ :: `=>b m`
be :: `=g b`
will :: `=b m`
-ϵ :: `=>f3 m`
-*ing* :: `=>f3 g`
-ϵ :: `=>f3 b`
-ϵ :: `=>f1 f3`
-ϵ :: `=>f2 f3`
laugh :: `=d f1`
jump :: `=d f2`

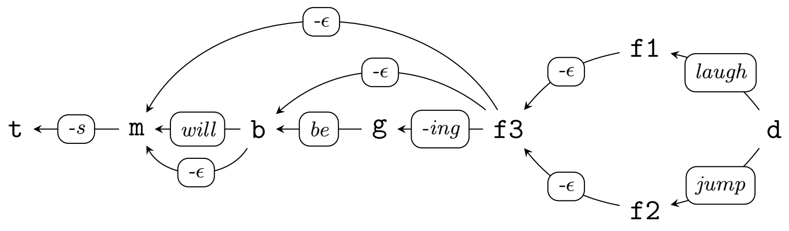


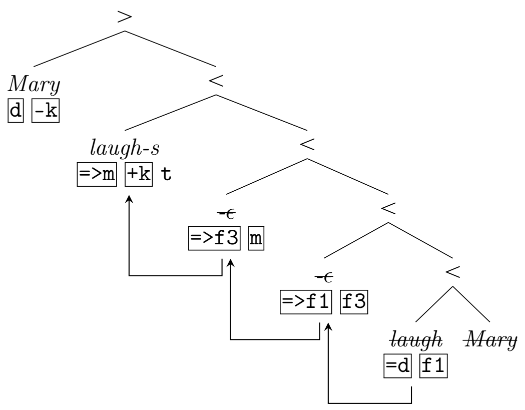
Figure 16: Fragment of the English auxiliary system

Consider -ϵ :: `=>f1 f2` and -ϵ :: `=>f1 f3`. As mentioned in Subsection 2.2, empty LIs introducing no specifiers encode the idea that some syntactic categories have a more limited distribution than others. In this case, however, it is safe to simplify the grammar by removing the silent LI and collapsing `f1` and `f2` into one category. Same goes for `f1` and `f3`. This is equivalent to saying that *laugh* and *jump* have the same syntactic distribution – which is true for this grammar.

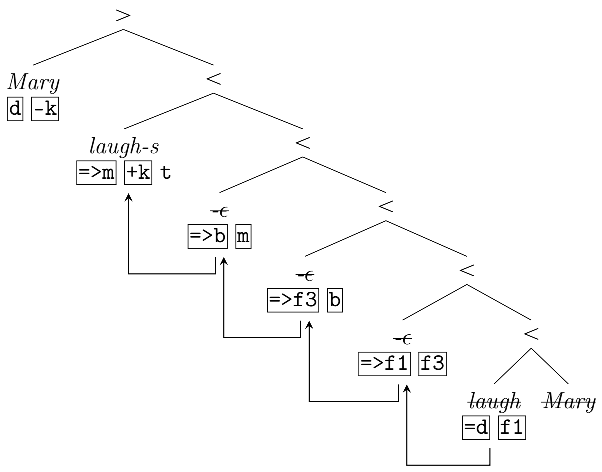
Other empty lexical items in this grammar should not be removed in the same way. For example, -ϵ :: `=>b m` is necessary to enforce the hierarchy of categories. Deleting it and collapsing `b` and `m` would create a cycle in the graph, allowing *will* to appear an arbitrary number of times in the same clause.

Furthermore, let us examine -ϵ :: `=>f3 m`. Its purpose is to generate sentences where the Tense suffix -*s* attaches directly to the lexical verb (Figure 17a). However, the same sentences can be generated via

`-`ϵ :: `=>f3 b` and `-`ϵ :: `=>b m` instead (Figure 17b). Thus, `-`ϵ :: `=>f3 m` can be safely removed from the grammar without any additional changes.



(a) Via `-`ϵ :: `=>f3 m`



(b) Via `-`ϵ :: `=>f3 b` and `-`ϵ :: `=>b m`

Figure 17: The structure of *Mary laugh-s* using Figure 16

Summarizing the above, we define two simplifying auxiliary operations to apply in tandem with batch decomposition:

- **Contraction**: remove an edge representing an empty lexical item and collapse the two vertices it connects into one. From the syntactic point of view, this means unifying two categories. This operation can lead to overgeneration and has to be constrained separately;
- **Deletion**: delete an edge representing an empty lexical item as long as there is an alternative path through the graph connecting its origin and destination. This operation is similar to lexical item decomposition but acknowledges LIs already present in the grammar rather than create new ones.

3.2 The algorithm

The discovery procedure can be characterized as a search problem, which has two components:

- a hypothesis space;
- a method of examining this space to find the best candidate.

In our case, the first component is supplied by the operations introduced in the previous section: we can now define the search space of candidate grammars as the set of MGs that can be obtained by applying lexical item decomposition and the simplifying operations to a given starting grammar.

The second component requires a metric – some way of selecting the better of two grammars on quantitative grounds, as in Chomsky's (1957) evaluation procedure. Linguistically motivated generalizations eliminate repetitions and remove redundancies in a grammar; intuitively, a good grammar is a small one. The measure has to take into account any elements that can differ between two grammars. For MGs they include the number of distinct symbols (syntactic categories, feature types, and the length of string components and syntactic feature bundles of all lexical items.[4]

That said, it is easy to construct an extremely short grammar that permits words to combine without any restrictions. Such a grammar would produce all sorts of ungrammatical constructions in addition to grammatical ones. To avoid this pitfall, in addition to measuring the size of the grammar, we need a way to eliminate overgenerating grammars.

One solution is provided by the Minimum Description Length principle (Rissanen 1978). The MDL metric is computed as the sum of the size of the grammar itself and of a given data corpus as encoded by the grammar. In this framework, an overgenerating grammar would be unable to compress the corpus efficiently, leading to a large corpus size. On the other hand, an grammar that makes few (or no) generalizations on the data would have a very large grammar size and small (or zero) corpus size. This tradeoff allows a balanced grammar to win over either extreme.

The evaluation measure used in this paper includes grammar size but not corpus size. Overgeneration is controlled separately by a heuristic: a grammar is ruled out if it generates any morphological words that differ (phonologically or syntactically) from those present in the original grammar.[5]

The number of decomposition steps available for a single input grammar can easily become very large, which makes checking all possibilities unfeasible. To mitigate this problem, the procedure for MG optimization developed here uses a heuristic known as *beam search* (Reddy 1977). This technique revolves around keeping track of a (relatively small) number of best hypotheses in the search space, expanding them to obtain new candidates, and discarding the rest. This parameter, *beam size*, is referred to as *bs*.

To provide a high-level description, grammar optimization proceeds as follows:

1. Initialize the set of candidate grammars as $\{G_0\}$, where G_0 is a "naive" grammar whose LIs are whole words;

[4]Using an encoding scheme adapted from Katzir (2014), the size of an MG can be calculated as follows. Each LI is treated as a sequence of symbols followed by a separator, and each syntactic feature counts as two symbols (corresponding to its type and name). Denoting the set of syntactic feature types as T, syntactic categories as $Base$, and symbols used in string components as Σ, the size of a grammar G is given by:

$$\underbrace{\sum_{s::\delta \,\in\, G} \big(|s| + 2 \times |\delta| + 1\big)}_{\text{total number of symbols}} \times \underbrace{\log_2(|\Sigma| + |T| + |Base| + 1)}_{\text{cost of encoding per symbol}}.$$

[5]For more discussion and comparison with MDL and other alternatives see (Ermolaeva 2021a, pp. 112–119).

2. For each candidate grammar, identify batches of lexical items that share some of their syntactic and/or phonological features;
3. Construct new grammars via batch decomposition;
4. Simplify the new grammars using contraction and deletion;
5. If a stopping condition is met, output the best known grammar according to the chosen metric. Otherwise, replace the candidate set with *bs* best new grammars and return to step 2.

A Python implementation of this optimization procedure, along with input grammars and code for a number of experiments, is available at https://github.com/mermolaeva/mg-optimizer.

4 Case study: lexical selection of prepositional phrases

An interesting test case for the procedure outlined above comes from lexical selection (l-selection, Pesetsky 1991) of prepositional phrases.

As reported by Merchant (2019), idiosyncratic selectional properties of most roots in English remain uniform across different realizations of the root. For example, the root $\sqrt{\text{RELI}}$ appears with a PP complement headed by the preposition *on* whether it is realized as the verb *rely* (1a), noun *reliance* (1b), or adjective *reliant* (1c). However, some roots break this pattern. There is a large class of cases, such as $\sqrt{\text{FEAR}}$, where the verb takes a direct object (2a) while the noun and adjective select a PP with *of* (2b, 2c). Furthermore, selectional properties of a number of roots vary across multiple realizations. This is the case with $\sqrt{\text{PRD}}$, which takes an *on*-PP as the verb *pride oneself* (3a), an *in*-PP as the noun *pride* (3b), and an *of*-PP as the adjective *proud* (3c).

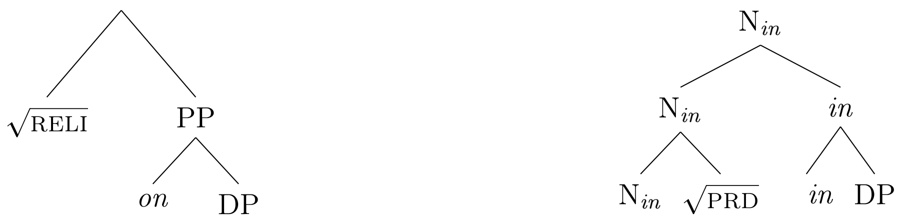
(1) a. They rely on oil.
    b. Their reliance on oil is well-known.
    c. They are reliant on oil.

(2) a. Abby fears dark spaces.
    b. Abby's fear of dark spaces is well known.
    c. Abby is fearful of dark spaces.

(3) a. She prides herself on/*in/*of her thoroughness.
    b. Her pride *on/in/*of her thoroughness is understandable.
    c. She is proud *on/*in/of her thoroughness.

(Merchant 2019, pp. 327, 329)

Regular examples like (1) provide an argument for roots being acategorial. They can be captured with an analysis like (18a), where the root itself selects for its complement, and the resulting structure is in turn selected by a head categorizing it as a noun, verb, or adjective. This analysis, however, does not work for roots with non-uniform selectional properties such as those in (2) and (3), where the choice of a complement alternates across realizations. To account for these, Merchant proposes a solution illustrated by 18b. Under this analysis, categorizing heads come with two pieces of information about selection: which roots they can combine with, and which PPs are compatible with those roots. For instance, the nominalizer N_{in} first selects for a compatible root (such as $\sqrt{\textsc{prd}}$) and then for an *in*-PP.



(a) Root selects for its complement PP (b) Nominalizer selects for the root and PP

Figure 18: Two possible structures for l-selection of PPs (adapted from Merchant 2019)

Root	V	N	A (-ful)
boast	of, about	of, about	of, about
disdain respect	DO	for	of
doubt	(DO)	of, about	of, about
fear neglect scorn	(DO)	of	of
hope	for	for	for

Table 1: V-N-A tuples and argument types

Will a similar analysis emerge in the process of grammar optimization? In order to focus on l-selection, let us consider a small grammar fragment featuring (mostly) concatenative morphology. Table 1 gives a sample of complete ⟨verb, noun, adjective⟩ triples selected from examples and database information presented in (Merchant 2019). The verbs and nouns don't carry any overt categorizing morphology, whereas the adjectives are formed with the suffix *-ful*. The sample includes roots with uniform complements (*boast*, *hope*); members of the large class alternating between direct objects for verbs and *of*-PPs for everything else (*doubt*, *fear*, *neglect*, *scorn*); and other non-uniform roots (*disdain*, *respect*, *doubt*). An MG over unsegmented words including realizations of these roots is given in Figure 19, with all lexical items listed in Figure 20. As before, we ignore non-concatenative morphology and assume a separate set of morphological rules which realize *haves* as *has*, *dos* as *does*, *willd* as *would*, etc.

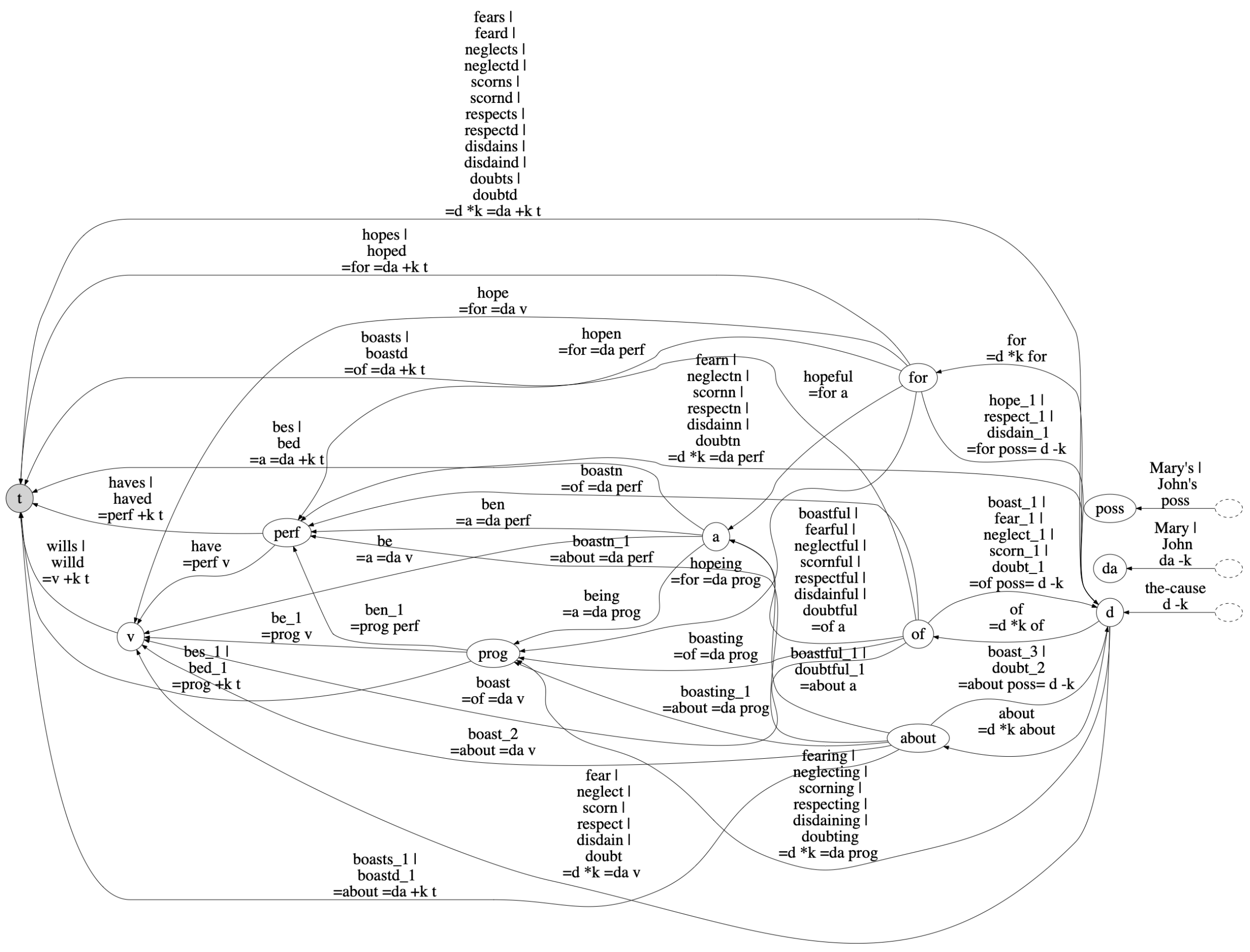

fears | feard | neglects | neglectd | scorns | scornd | respects | respectd | disdains | disdaind | doubts | doubtd =d *k =da +k t
hopes | hoped =for =da +k t
hope =for =da v
boasts | boastd =of =da +k t
hopen =for =da perf
fearn | neglectn | scornn | respectn | disdainn | doubtn =d *k =da perf
hopeful =for a
for =d *k for
hope_1 | respect_1 | disdain_1 =for poss= d -k
bes | bed =a =da +k t
boastn =of =da perf
haves | haved =perf +k t
ben =a =da perf
boastful | fearful | neglectful | scornful | respectful | disdainful | doubtful =of a
boast_1 | fear_1 | neglect_1 | scorn_1 | doubt_1 =of poss= d -k
Mary's | John's poss
Mary | John da -k
the-cause d -k
wills | willd =v +k t
have =perf v
be =a =da v
boastn_1 =about =da perf
hopeing =for =da prog
being =a =da prog
of =d *k of
be_1 =prog v
ben_1 =prog perf
boasting =of =da prog
bes_1 | bed_1 =prog +k t
boast =of =da v
boastful_1 | doubtful_1 =about a
boasting_1 =about =da prog
boast_3 | doubt_2 =about poss= d -k
about =d *k about
boast_2 =about =da v
fearing | neglecting | scorning | respecting | disdaining | doubting =d *k =da prog
fear | neglect | scorn | respect | disdain | doubt =d *k =da v
boasts_1 | boastd_1 =about =da +k t
t
perf
v
prog
a
for
of
about
poss
da
d

Figure 19: l-selection before optimization

John	::	`da -k`
John's	::	`poss`
Mary	::	`da -k`
Mary's	::	`poss`
the cause	::	`d -k`
about	::	`=d *k about`
for	::	`=d *k for`
of	::	`=d *k of`

(a) Nouns and prepositions

willd	::	`=v +k t`			
wills	::	`=v +k t`			
have	::	`=perf v`			
haved	::	`=perf +k t`			
haves	::	`=perf +k t`	*be*	::	`=a =da v`
be	::	`=prog v`	*bed*	::	`=a =da +k t`
bed	::	`=prog +k t`	*being*	::	`=a =da prog`
ben	::	`=prog perf`	*ben*	::	`=a =da perf`
bes	::	`=prog +k t`	*bes*	::	`=a =da +k t`

(b) Auxiliaries and copulas

boast	::	`=about =da v`	*boast*	::	`=of =da v`
boastd	::	`=about =da +k t`	*boastd*	::	`=of =da +k t`
boasting	::	`=about =da prog`	*boasting*	::	`=of =da prog`
boastn	::	`=about =da perf`	*boastn*	::	`=of =da perf`
boasts	::	`=about =da +k t`	*boasts*	::	`=of =da +k t`
boast	::	`=about poss= d -k`	*boast*	::	`=of poss= d -k`
boastful	::	`=about a`	*boastful*	::	`=of a`

(c) *boast*

disdain	::	`=d *k =da v`
disdaind	::	`=d *k =da +k t`
disdaining	::	`=d *k =da prog`
disdainn	::	`=d *k =da perf`
disdains	::	`=d *k =da +k t`
disdain	::	`=for poss= d -k`
disdainful	::	`=of a`
respect	::	...

(d) *disdain*, *respect*

doubt	::	`=d *k =da v`
doubtd	::	`=d *k =da +k t`
doubting	::	`=d *k =da prog`
doubtn	::	`=d *k =da perf`
doubts	::	`=d *k =da +k t`
doubt	::	`=about poss= d -k`
doubtful	::	`=about a`
doubt	::	`=of poss= d -k`
doubtful	::	`=of a`

(e) *doubt*

fear	::	`=d *k =da v`
feard	::	`=d *k =da +k t`
fearing	::	`=d *k =da prog`
fearn	::	`=d *k =da perf`
fears	::	`=d *k =da +k t`
fear	::	`=of poss= d -k`
fearful	::	`=of a`
neglect	::	...
scorn	::	...

(f) *fear*, *neglect*, *scorn*

hope	::	`=for =da v`
hoped	::	`=for =da +k t`
hopeing	::	`=for =da prog`
hopen	::	`=for =da perf`
hopes	::	`=for =da +k t`
hope	::	`=for poss= d -k`
hopeful	::	`=for a`

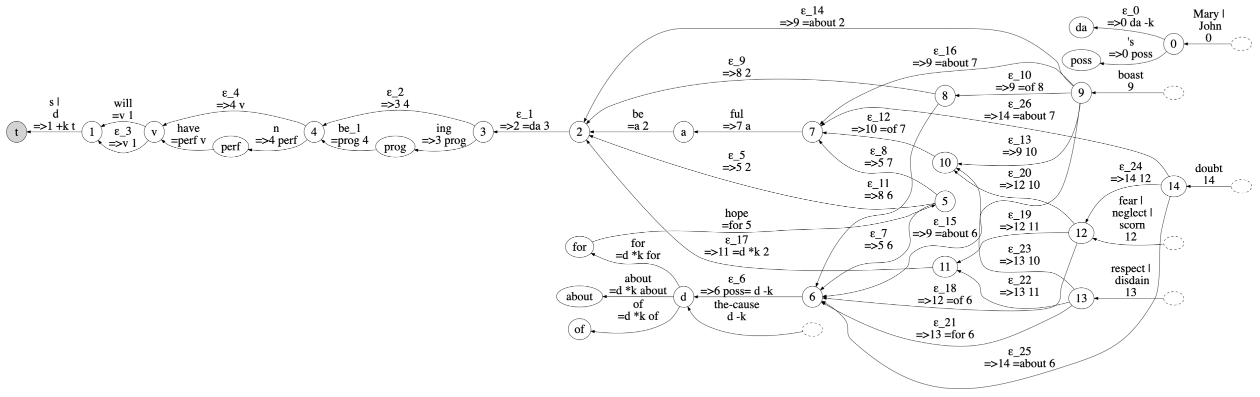
(g) *hope*

Figure 20: Lexical items of Figure 19

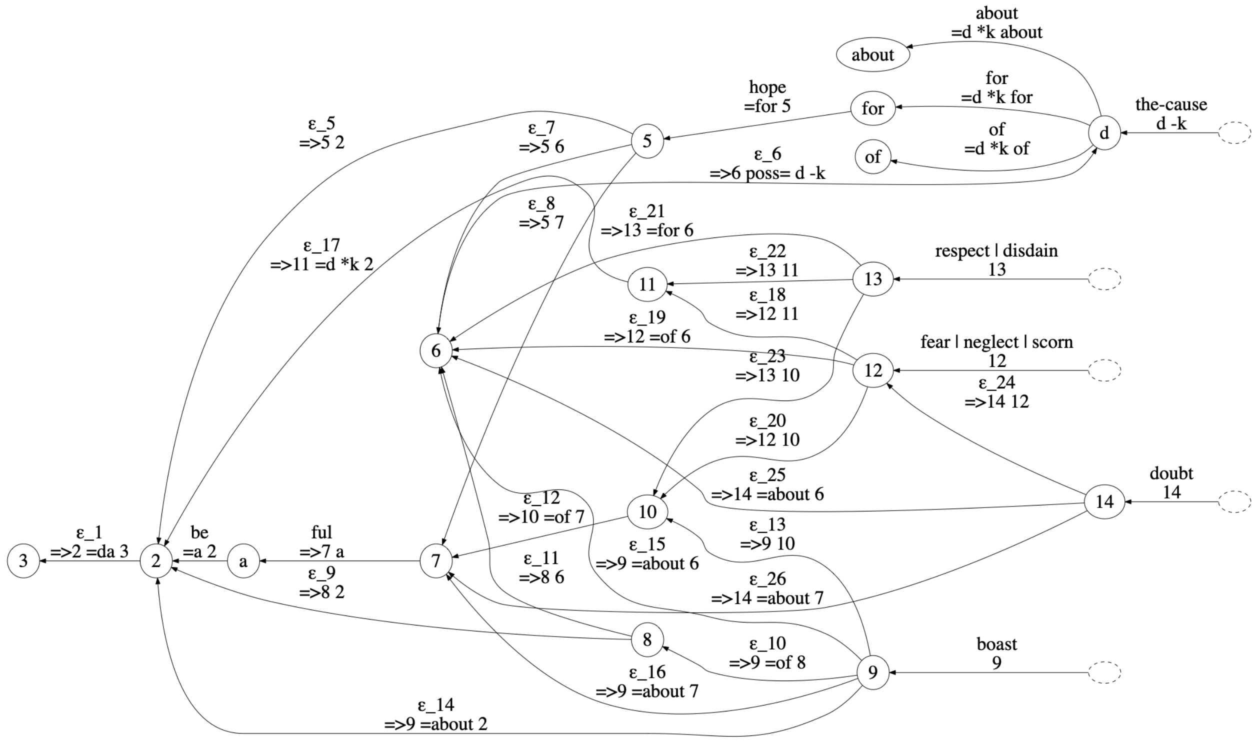
The language generated by this grammar is infinite, since complex DPs can, in turn, be selected by verbs or prepositions. Some sentences from this language are given below:

Mary hopes for the cause;
Mary haves ben respectful of the cause;
Mary bes boasting about John's fear of the cause;
Mary wills be disdainful of the cause;
Mary doubts John's respect for the cause;
Mary bes scornful of John's neglect of Mary's hope for the cause.

Running the optimization procedure on this input with a large beam size ($bs = 500$) yields the grammar given in Figure 21 and Figure 22.



(a) Full graph



(b) Close-up of verbs and categorizing heads

Figure 21: l-selection after optimization, $bs = 500$

John ::	`0`
Mary ::	`0`
-*'s* ::	`=>0 poss`
-ϵ ::	`=>0 da -k`
the cause ::	`d -k`
about ::	`=d *k about`
for ::	`=d *k for`
of ::	`=d *k of`

(a) Nouns and prepositions

-*d* ::	`=>1 +k t`
-*s* ::	`=>1 +k t`
will ::	`=v 1`
-ϵ ::	`=>v 1`
have ::	`=perf v`
-*n* ::	`=>4 perf`
-ϵ ::	`=>4 v`
be ::	`=prog 4`
-*ing* ::	`=>3 prog`
-ϵ ::	`=>3 4`
be ::	`=a 2`
-ϵ ::	`=>a =da 3`

(b) Auxiliaries and copulas

boast ::	`9`
disdain ::	`13`
doubt ::	`14`
fear ::	`12`
hope ::	`=for 5`
neglect ::	`12`
respect ::	`13`
scorn ::	`12`

(c) Lexical verbs

		-ϵ :: `=>6 poss= d -k`	-ϵ :: `=>9 =of 8`
		-ϵ :: `=>5 6`	-ϵ :: `=>9 10`
-*ful* :: `=>7 a`	-ϵ :: `=>2 =da 3`	-ϵ :: `=>8 6`	-ϵ :: `=>12 11`
-ϵ :: `=>5 7`	-ϵ :: `=>11 =d *k 2`	-ϵ :: `=>9 =about 6`	-ϵ :: `=>12 10`
-ϵ :: `=>9 =about 7`	-ϵ :: `=>5 2`	-ϵ :: `=>12 =of 6`	-ϵ :: `=>13 11`
-ϵ :: `=>10 =of 7`	-ϵ :: `=>8 2`	-ϵ :: `=>13 =for 6`	-ϵ :: `=>13 10`
-ϵ :: `=>14 =about 7`	-ϵ :: `=>9 =about 2`	-ϵ :: `=>14 =about 6`	-ϵ :: `=>14 12`

(d) Categorizing heads

Figure 22: Lexical items of Figure 21

The differences between Figure 19 and Figure 21 are extensive, as most of the original LIs have undergone decomposition. This is showcased in particular by the auxiliary system(22b). All verbal paradigms have been decomposed, forming a single set of suffixes (-*ing*, -*n*, -*s*, -*d*) compatible with all verb stems. Next, hierarchical relations between categories are enforced by empty LIs, which form an implicit hierarchy of projections: `3` $<$ `4` $<$ `v` $<$ `1` $<$ `t`. This ensures, for instance, that an expression headed by the auxiliary *be* can be selected by *will*, but not the other way around; whereas the Tense heads -*s* and -*d* will take either. In the nominal domain (22a), the original feature sequences associated with animate DPs (`da -k` and `poss`) have been split off into empty LIs, while proper nouns are left with a new feature, `0`.

Apart from these changes , this grammar features extensive modifications to how roots are categorized. All roots except *hope* have been reduced to a single category (22c). Among the remaining LIs, three are easily identifiable as categorizing heads. The verbalizer -ϵ :: `=>2 =da 3` is responsible for the verb's external argument, the nominalizer -ϵ :: `=>6 poss= d -k` merges in a possessive phrase such as *John's* and assigns the entire structure the typical DP feature sequence of `d -k`, and the adjectivizer *-ful* :: `=>7 a` provides the overt affix and the category `a`.

The rest of LIs in 22d are empty heads selecting first one of the new categories; some of them additionally introduce a PP headed by a specific preposition. To form a clearer picture of what role each of them plays, we need to consider what each of the new syntactic features stands for. A brief summary is provided in Table 2.

Category	Interpretation
`5`	*hope*
`9`	*boast*
`12`	*fear*, *neglect*, *scorn*
`13`	*disdain*, *respect*
`14`	*doubt*
`2`	compatible with the verbalizer -ϵ :: `=>2 =da 3`
`6`	compatible with the nominalizer -ϵ :: `=>6 poss= d -k`
`7`	compatible with the adjectivizer *-ful* :: `=>7 a`
`8`	verbal and nominal *boast* with an *of*-PP argument
`10`	adjectives with an *of* PP argument
`11`	verbs with a direct object

Table 2: Interpretation of features in Figure 21

The first major group of features (`5`, `9`, `12`, `13`, `14`) stands for roots. Reducing open-class items to shorter feature bundles is an expected optimization strategy. Roots with identical selectional properties (*fear*, *neglect*, *scorn*; *disdain*, *respect*) have been assigned the same category. Most of the roots don't select their arguments directly, instead relying on whatever LIs select them to provide the necessary features. The only exception is *hope* :: `=for 5`. This root uniformly selects *for*-PPs across all realizations, and is the only one in the sample to do so – which makes it a suboptimal decision to decompose it further. That said, not all cases of uniform l-selection are treated in the same way. The other such root (*boast* :: `9`) carries a single category, and its selectional properties are handled by other LIs. One plausible explanation is that the dataset is very small, and *every* root is effectively treated as idiosyncratic. With a larger grammar, we might be able to see a more uniform treatment of frequently occurring patterns.

The second group (`2`, `6`, `7`) is responsible for categorization. Any expression of category `2`, whatever LI it is headed by, is compatible with (i.e. can be selected by) the verbalizing head. This includes verbs which take a direct object, as well as those taking a PP argument, such as *boast* and *hope*. Similarly, `6` represents nouns, and `7` adjectives. Finally, the third group (`8`, `10`, `11`) picks up on more minor patterns, which are nevertheless quantitatively worthy of a separate category.

With this in mind, any valid configuration of a root, category, and argument type can be assembled with a specific combination of silent LIs, summarized in Table 3. Note that there are multiple heads selecting PPs, rather than only one per preposition type. Under Merchant's (2019) analysis, each categorizing head is annotated with its argument type and a list of compatible roots. Minimalist grammars encode this by having a separate lexical item for each option.[6]

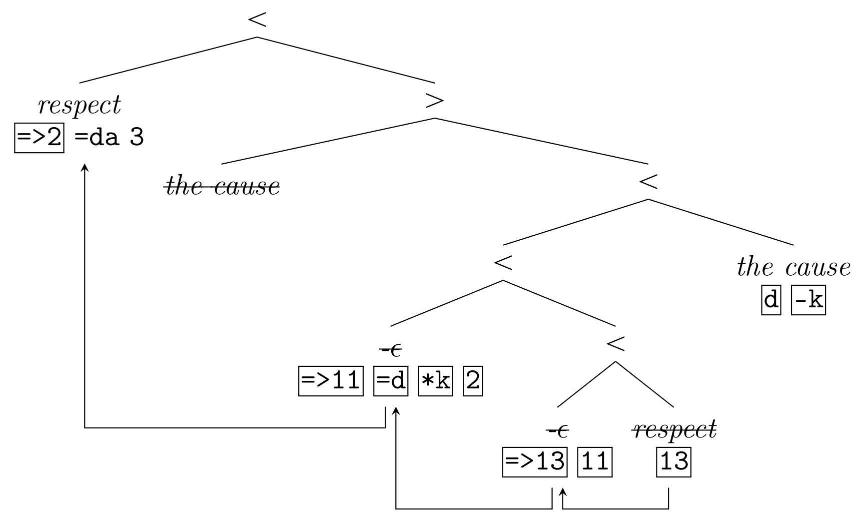
Similarly, silent LIs are the MGs' way of encoding distributional similarities and differences between roots. For example, *doubt* can take dependents of the same type as *fear*/*neglect*/*scorn*, and additionally can select an *about*-PP when realized as a noun or adjective – unlike *fear*-type roots. This difference warrants a separate category `14` for *doubt*. However, the presence of -ϵ :: `=>14 12` allows *doubt* to piggy-back on the system in place for *fear*-type verbs (of category `12`) for the shared part of its selectional options. Essentially, this LI encodes the statement that *doubt* can be found in all contexts of *fear*, as well as some of its own.

For a concrete illustration, trees in Figure 23 demonstrate how the optimized grammar derives the three realizations of *respect* and their non-uniform selectional properties.

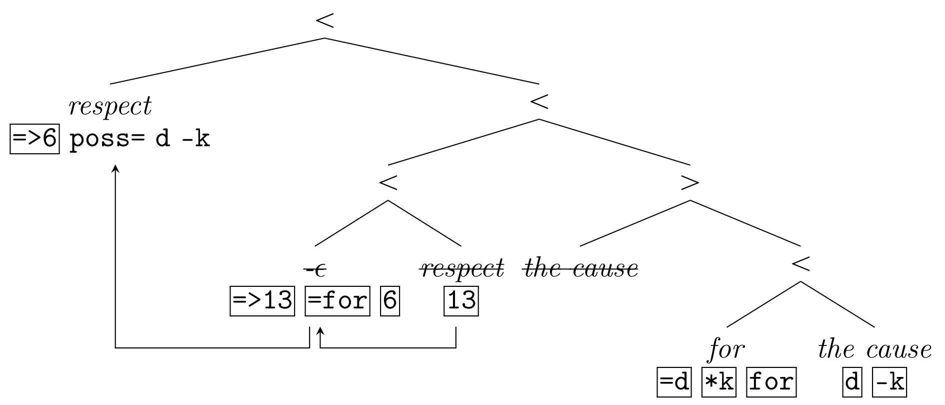
[6]A good question to ask is whether we could leverage the idea of a list to inform our encoding scheme. For example, it may be possible to encourage multiple LIs which only differ in one syntactic feature by recording everything they share only once, making them cheaper to encode.

Root	Argument	Category	Implementation in LIs
boast	about	V	*boast* :: `9`, -ϵ :: `=>9 =about 2`
		N	*boast* :: `9`, -ϵ :: `=>9 =about 6`
		A	*boast* :: `9`, -ϵ :: `=>9 =about 7`
	of	V	*boast* :: `9`, -ϵ :: `=>9 =of 8`, -ϵ :: `=>8 2`
		N	*boast* :: `9`, -ϵ :: `=>9 =of 8`, -ϵ :: `=>8 6`
		A	*boast* :: `9`, -ϵ :: `=>9 10`, -ϵ :: `=>10 =of 7`
disdain	DO	V	*disdain* :: `13`, -ϵ :: `=>13 11`, -ϵ :: `=>11 =d *k 2`
	for	N	*disdain* :: `13`, -ϵ :: `=>13 =for 6`
	of	A	*disdain* :: `13`, -ϵ :: `=>13 10`, -ϵ :: `=>10 =of 7`
respect	DO	V	*respect* :: `13`, -ϵ :: `=>13 11`, -ϵ :: `=>11 =d *k 2`
	for	N	*respect* :: `13`, -ϵ :: `=>13 =for 6`
	of	A	*respect* :: `13`, -ϵ :: `=>13 10`, -ϵ :: `=>10 =of 7`
doubt	DO	V	*doubt* :: `14`, -ϵ :: `=>14 12`, -ϵ :: `=>12 11`, -ϵ :: `=>11 =d *k 2`
	about	N	*doubt* :: `14`, -ϵ :: `=>14 =about 6`
		A	*doubt* :: `14`, -ϵ :: `=>14 =about 7`
	of	N	*doubt* :: `14`, -ϵ :: `=>14 12`, -ϵ :: `=>12 =of 6`
		A	*doubt* :: `14`, -ϵ :: `=>14 12`, -ϵ :: `=>12 10`, -ϵ :: `=>10 =of 7`
fear	DO	V	*fear* :: `12`, -ϵ :: `=>12 11`, -ϵ :: `=>11 =d *k 2`
	of	N	*fear* :: `12`, -ϵ :: `=>12 =of 6`
	of	A	*fear* :: `12`, -ϵ :: `=>12 10`, -ϵ :: `=>10 =of 7`
neglect	DO	V	*neglect* :: `12`, -ϵ :: `=>12 11`, -ϵ :: `=>11 =d *k 2`
	of	N	*neglect* :: `12`, -ϵ :: `=>12 =of 6`
	of	A	*neglect* :: `12`, -ϵ :: `=>12 10`, -ϵ :: `=>10 =of 7`
scorn	DO	V	*scorn* :: `12`, -ϵ :: `=>12 11`, -ϵ :: `=>11 =d *k 2`
	of	N	*scorn* :: `12`, -ϵ :: `=>12 =of 6`
	of	A	*scorn* :: `12`, -ϵ :: `=>12 10`, -ϵ :: `=>10 =of 7`
hope	for	V	*hope* :: `=for 5`, -ϵ :: `=>5 2`
	for	N	*hope* :: `=for 5`, -ϵ :: `=>5 6`
	for	A	*hope* :: `=for 5`, -ϵ :: `=>5 7`

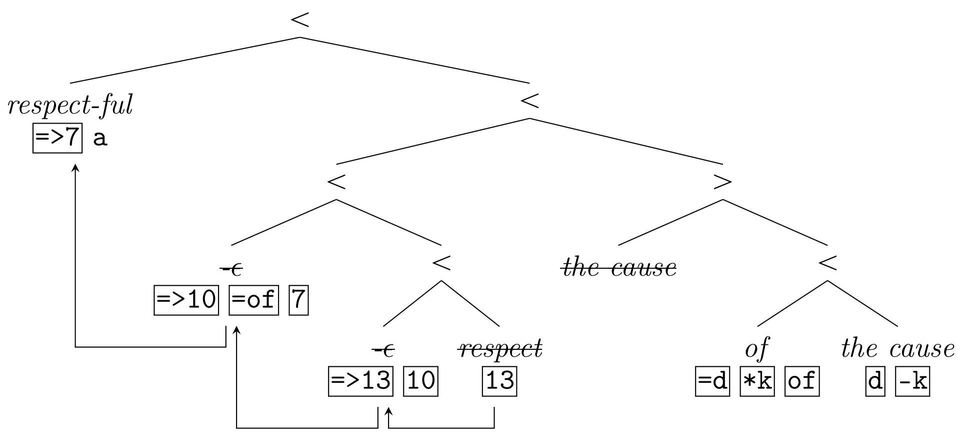
Table 3: Verbs and categorizing heads in Figure 21



(a) Verb



(b) Noun



(c) Adjective

Figure 23: Selectional variability of *respect* (using Figure 21)

Some properties of the optimized grammar stem from those of the formalism. Selection in (this version of) MGs is symmetric, which means roots cannot be truly acategorial; as long as heads selecting them are compatible with some roots but not all of them, the roots themselves must carry different category features to enable this distinction. Similarly, each categorizing head is represented as a combination of multiple lexical items, where in a theoretical work it might be a single head assigned different surface realizations by additional vocabulary insertion rules. However, modulo these (largely notational) differences, the emerging MG is fundamentally similar to the proposal of (Merchant 2019). The algorithm has arrived at the conclusion that (most) roots don't select, whereas categorizing heads do.

5 Discussion

In this paper, I take a step towards formalizing the intuition behind linguistic generalizations. The result of this effort is a procedure for refining and optimizing descriptions of natural language syntax expressed as minimalist grammars.

Let us consider some general patterns and strategies repeatedly used by the optimization procedure. There is a strong tendency for assigning very short feature bundles to roots and pushing their phonological and/or syntactic differences into newly created affixes. In particular, the algorithm has made extensive use of the fact that lexical item decomposition can derive phonologically empty heads. The fact that it has converged on a grammar similar to that proposed in the literature is encouraging but not especially surprising, as this is beneficial from the quantitative perspective: associating a large number of string components with a short feature bundle allows the grammar to encode each string component and each feature bundle only once, avoiding redundancy.

Taken to the logical conclusion, this approach leaves roots with a single category feature (the shortest possible feature bundle) and creates empty heads to supply their syntactic properties. This is consistent with the idea of having acategorial roots and categorizing heads, Distributed Morphology-style. In a larger grammar, we can expect this strategy to be pursued especially aggressively with open-class items, since they would make up the bulk of words with identical syntactic distribution.

Two broad directions of future work are of special interest for this project. The first goal is to improve the evaluation measure, including the way the grammar is encoded. A grammar formalism and a quantitative metric by themselves do not guarantee desirable results; for example, see de Marcken (1995) on stochastic context-free grammars misinterpreting English phrase structure. The encoding scheme used in this paper was chosen for its conceptual simplicity; it may be possible to devise a better one – for example, to encourage higher level generalizations by lowering the cost of reusing existing syntactic feature bundles.

The second direction, interleaved with the first one, involves lifting the limitations discussed in Subsection 2.2. This includes, in particular, moving away from the somewhat simplistic definition of lexical items over strings of phonological or even orthographic segments. One alternative is to have LIs carry semantic features instead of string components. Morphological words would then become sequences of *semantic* feature bundles, which would then be manipulated by a separate “morphology proper” module mapping them to strings. defining lexical items over Other desirable additions to the formalism include agreement and generalized head movement.

References

D. Adger. A minimalist theory of feature structure. *Features: Perspectives on a key notion in linguistics*, pages 185–218, 2010.

K. Arregi and A. Pietraszko. Generalized head movement. *Proceedings of the Linguistic Society of America*, 3(1):1–15, 2018.

M. Brody. Mirror theory: Syntactic representation in perfect syntax. *Linguistic Inquiry*, 31(1):29–56, 2000.

N. Chomsky. *Syntactic structures.* Mouton, The Hague, 1957.

N. Chomsky. Aspects of the theory of syntax. *MIT Press*, 1965.

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

N. Chomsky. Minimalist Inquiries: the framework. In R. Martin, D. Michaels, and J. Uriagereka, editors, *Step by Step: Essays on Minimalist Syntax in Honor of Howard Lasnik*, pages 89–156. MIT Press, Cambridge, MA, 2000.

A. Clark. Computational learning of syntax. *Annual Review of Linguistics*, 3:107–123, 2017.

A. Clark and R. Eyraud. Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research*, 8(Aug):1725–1745, 2007.

C. de la Higuera. *Grammatical inference: learning automata and grammars.* Cambridge University Press, 2010.

C. de Marcken. Lexical heads, phrase structure and the induction of grammar. In *Third Workshop on Very Large Corpora*, 1995. URL https://www.aclweb.org/anthology/W95-0102.

D. Embick and A. Marantz. Architecture and blocking. *Linguistic inquiry*, 39(1):1–53, 2008.

M. Ermolaeva. Morphological agreement in minimalist grammars. In *International Conference on Formal Grammar*, pages 20–36. Springer, 2018.

M. Ermolaeva. *Learning syntax via decomposition.* PhD thesis, University of Chicago, 2021a.

M. Ermolaeva. Deconstructing syntactic generalizations with minimalist grammars. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 435–444, 2021b.

M. Ermolaeva and G. M. Kobele. Agree as information transmission over dependencies. *Syntax*, 2022.

M. Fowlie. Order and optionality: Minimalist grammars with adjunction. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pages 12–20, 2013.

T. Graf. *Local and Transderivational Constraints in Syntax and Semantics.* PhD thesis, UCLA, 2013.

S. Indurkhya. Automatic inference of minimalist grammars using an SMT-solver. *arXiv preprint arXiv:1905.02869*, 2019.

R. Katzir. A cognitively plausible model for grammar induction. *Journal of Language Modelling*, 2, 2014.

G. M. Kobele. Formalizing mirror theory. *Grammars*, 5(3):177–221, 2002.

G. M. Kobele. Minimalist grammars and decomposition. In K. K. Grohmann and E. Leivada, editors, *The Cambridge Handbook of Minimalism.* Cambridge University Press, Cambridge, to appear.

S. Laszakovits. Case theory in minimalist grammars. In *International Conference on Formal Grammar*, pages 37–61. Springer, 2018.

A. Marantz. No escape from syntax: Don't try morphological analysis in the privacy of your own lexicon. *University of Pennsylvania working papers in linguistics*, 4(2):14, 1997.

J. Merchant. Roots don't select, categorial heads do: lexical-selection of PPs may vary by category. *The Linguistic Review*, 36(3):325–341, 2019.

D. Pesetsky. Zero syntax. vol. 2: Infinitives, 1991.

J.-Y. Pollock. Verb movement, universal grammar, and the structure of ip. *Linguistic inquiry*, 20(3):365–424, 1989.

R. Reddy. Speech understanding systems: summary of results of the five-year research effort at Carnegie-Mellon University., 1977.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

E. P. Stabler. Derivational minimalism. In C. Retoré, editor, *Logical Aspects of Computational Linguistics: First International Conference, LACL '96 Nancy, France, September 23–25, 1996 Selected Papers*, pages 68–95. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

E. P. Stabler. Recognizing head movement. In *Proceedings of the 4th International Conference on Logical Aspects of Computational Linguistics*, LACL '01, pages 245–260, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-42273-0.

R. Yoshinaka. Efficient learning of multiple context-free languages with multidimensional substitutability from positive data. *Theoretical Computer Science*, 412(19):1821–1831, 2011.