

Evaluating syntactic proposals using minimalist grammars and minimum description length*

Marina Ermolaeva
Lomonosov Moscow State University

ABSTRACT

Many patterns found in natural language syntax have multiple possible explanations or structural descriptions. Even within the currently dominant Minimalist framework (Chomsky 1995, 2000), it is not uncommon to encounter multiple types of analyses for the same phenomenon proposed in the literature. A natural question, then, is whether one could evaluate and compare syntactic proposals from a quantitative point of view. In this paper, we show how an evaluation measure inspired by the minimum description length principle (Rissanen 1978) can be used to compare accounts of syntactic phenomena implemented as minimalist grammars (Stabler 1997), and how arguments for and against this kind of analysis translate into quantitative differences.

Keywords: syntax, evaluation measure, minimum description length, minimalist grammars, double object construction

INTRODUCTION

1

Even within the same framework, different proposals often seem equally capable of capturing observed linguistic phenomena, which creates a need for an additional criterion to choose between them.

*This paper is based on some parts of the author's PhD thesis (Ermolaeva 2021).

This idea is prominent in early generative grammar. An *evaluation procedure* – a method of determining which of the two given grammars is better, given a corpus of data – is discussed in Chomsky 1957. It makes another appearance in Chomsky 1965, where an *explanatory theory* of language is defined as one capable of selecting a descriptively adequate grammar based on linguistic data. The components of such a theory mirror those of an *acquisition model*, i.e. how a child learns a language, and are listed below:

- 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)

The last requirement (v) is described as being twofold: it calls for a formal *evaluation measure*, some sort of quantitative indication of how good a grammar is, but also demands that the class of possible grammars be small enough so the evaluation measure can realistically choose between them. In this framework, a precise and rich definition of “generative grammar” serves to tighten the class of grammars. However, the theory still permits multiple grammars compatible with the same data set; the choice of grammar is under-determined by the language data alone. This is where the evaluation measure comes in: the correct grammar is the highest-valued one among those that describe the data correctly. Of course, exactly how to construct a reasonable evaluation measure is a major issue by itself. Chomsky and Halle (1968) make some specific steps in this direction (for phonological rules), including a proposal of an evaluation procedure based on rule length measured in symbols.

Chomsky’s later work takes the idea of restricting what counts as a candidate grammar much further. By Chomsky 1986, the description of a grammar has shifted away from rule systems and is split into two components: an innate universal system of principles and parameters and a language-specific lexicon of items defined by their

phonological form and semantic properties, with the former getting most of the attention. Assuming a finite number of principles, parameters, and parameter values, the number of possible languages (apart from the lexicon) is also finite. This move sharply reduces the role of the evaluation measure or even dispenses with it altogether, as long as the universal grammar can be designed to permit only a single grammar compatible with the data.¹ The most recent and currently dominant iteration of generative grammar, the Minimalist Program (Chomsky 1995, 2000), continues this trend. Much of the system is assumed to be universal and innate, leaving no need or place for an evaluation measure; and language-specific properties that must be learned are largely shifted into the features of lexical items.

To summarize, the framework of Chomsky (1965) allows for multiple descriptions of a given language, one of which is the correct grammar, and these descriptions can be compared based on some quantitative measure. On the other hand, another framework he proposed (Chomsky 1986), as well as his later work, allow for a small number of descriptions of a given language, or even a single one; the correct grammar follows from the formal properties of the system and the language data. This stance can be considered a special case of the previous one, where the set of candidate grammars is made sufficiently small to eliminate the need for an evaluation measure.

Even though these two approaches are often thought of as mutually exclusive, they can be reconciled. Goldsmith (2011) and Katzir (2014) argue in favor of an evaluation measure based on the principle of minimum description length (MDL, Rissanen 1978), which takes into account both how good a grammar is by itself and how well it fits the data. MDL is compatible with any theory of universal grammar – as long as the grammars permitted by it are capable of *parsing*, or assigning structural descriptions to sentences as per (iv), and their description length can be compared. In line with these ideas, we combine the learning focus of (Chomsky 1965) with the simplifying developments

¹The *strong learning* approach of Clark 2013, 2015 can be thought of as a formalization of this idea. For each set of strings, it requires the existence of a unique description called the *canonical grammar*. A strong learning algorithm is required to converge to this target grammar for each (formal) language.

of Minimalism, applying an evaluation measure to Minimalist lexical items.

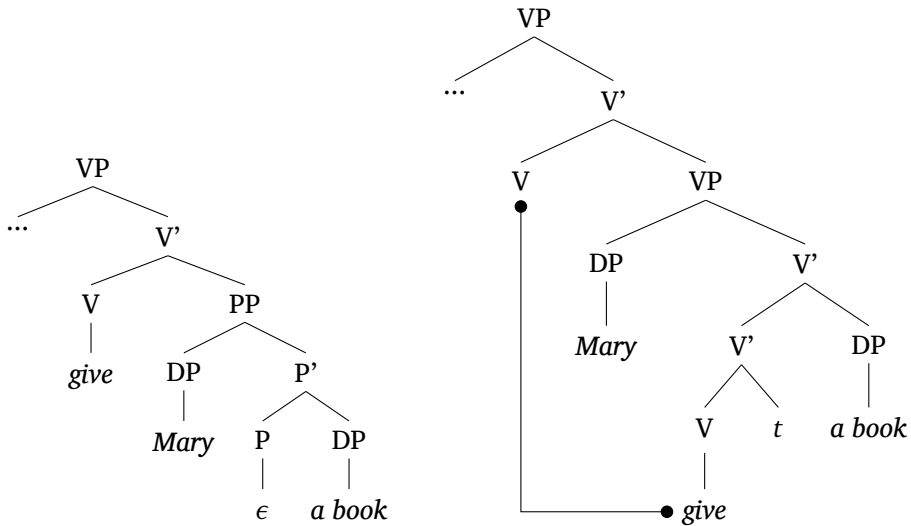
One major issue we have to tackle right from the start is that of formalization. Marr (1982) distinguishes between three levels of description of complex cognitive systems, including language:

- Computational: abstract specification of what the system computes;
- Algorithmic: structures representing the data and algorithms that manipulate them;
- Implementational: concrete realization of the algorithms in the hardware or wetware.

Johnson (2017) considers linguistic theories to be computational-level, while Peacocke (1986) places them at a “level 1.5”, between the computational and algorithmic level. Syntactic literature in particular tends to abstract away from algorithmic-level details such as full specifications of lexical items involved in derivations or syntactic features being checked by each application of a structure-building operation. At the same time, differences between competing analyses of the same phenomenon seem to fall closer to the algorithmic level.

For a specific example, consider the double object construction (e.g. *John gave Mary a book*) in English (Figure 1). Any analysis of a syntactic phenomenon encodes two kinds of information: relatively theory-neutral, high-level facts that directly follow from the data, such as relations between words based on argument structure and linear order; and a proposed explanation of these facts – for instance, a specific configuration of lexical items constructed by structure-building operations. Descriptively, ditransitive verbs such as *give* appear in active sentences with three arguments: a subject, a direct object, and an indirect object. This is (apparently) non-controversial. On the lower level,² disregarding the subject, one option is to combine the two internal arguments together and have the verb select the resulting constituent as its complement (Figure 1a). The arguments are described in

²Work concerning these structures also tends to assume and try to explain a connection between them and prepositional constructions, as in *John gave a book to Mary*. This too is a nontrivial analytical choice; see Goldsmith 1980. A sketch of comparison between grammars along this dimension is given in Appendix A.



(a) Null P (adapted from Pesetsky 1996) (b) VP-shells (adapted from Larson 1988)

Figure 1: The double object construction

terms of Williams’ (1975) “small clauses” or taken to be connected by a silent preposition-like element (Kayne 1984; Pesetsky 1996; Harley 2002; Harley and Jung 2015). The alternative is to have the verb form a constituent with one of its internal arguments and then select the other one (Figure 1b). This option gives rise to VP-shells (Larson 1988) and analyses inspired by them (Kawakami 2018).

Existing treatments of the double object construction generally fall into one of the two categories mentioned above, as there are only so many conceivable ways to form a binary-branching structure containing a verb and two arguments. That said, the abundance of recent literature on the topic indicates that this is far from a closed issue.

Given a disagreement in the literature over a specific linguistic puzzle, how can the competing solutions be compared in terms of Chomsky’s evaluation procedure? In order to take on this question, one needs to capture precisely what makes them different. This requires formalizing syntactic proposals at the algorithmic level, expressing them as a clearly defined set of building blocks and rules for putting them together. This paper adopts the formalism known as *minimalist grammars* (MGs), introduced by Stabler (1997). On the one hand,

minimalist grammars were expressly designed as an implementation of Chomsky’s Minimalist Program³ and offer a way to state analyses of syntactic phenomena in terms familiar to a linguist: lexical items defined by features and structure-building operations that combine them.⁴ On the other hand, they are explicit in spelling out assumptions about syntactic units and operations, and their formal properties – such as the complexity of string languages they generate and relation to other grammar formalisms – are relatively well understood.

This paper is structured as follows. Section 2 discusses the MDL principle, along with a toy example to demonstrate it in action. Section 3 provides a semi-formal, example-driven description of minimalist grammars. Section 4 builds on the previous sections to outline an encoding scheme for MGs and show how various intuitive notions translate into MDL values. In Section 5 we move away from toy examples and look at how MDL and MGs can be used to approach the problem of the double object construction. Finally, Section 6 offers some higher-level discussion and indicates some directions for future work.

2 THE MINIMUM DESCRIPTION LENGTH PRINCIPLE

Minimum description length (Rissanen 1978) is a principle for selecting a model to explain a dataset, which takes into account the simplic-

³The choice of capitalization – uppercase for “Minimalist Program” and lowercase for “minimalist grammars” – follows the sources that introduced these terms, Chomsky (1995) and Stabler (1997), respectively.

⁴Why minimalist grammars? A fully fleshed-out formalism is necessary to compute a quantitative measure such as MDL for each proposal in a self-contained way, independently from other candidates. That said, *which* formalism to use is a nontrivial decision, as any choice involves a tradeoff between conceptual simplicity and faithfulness to the original theoretical proposals. MGs do appear to diverge from mainstream Minimalist syntax with respect to the feature calculus, implementation of movement, locality, and other issues. However, as discussed in depth by Graf (2013, pp. 96–125), many of these apparent points of disagreement are a matter of convenience rather than an integral part of the formalism. We will briefly return to the problem of choosing a formalism in Section 6.

ity of both the model itself and the explanation of the dataset it offers. In the MDL framework, the best grammar to describe a corpus is the one that minimizes the sum of the following:

- the length of the grammar, measured in bits;
- the length of the description assigned by the grammar to the corpus, measured in bits.

Within linguistics, MDL has been used as a method of comparing candidate analyses of a given dataset, for example, for induction of phonological constraints (Rasin and Katzir 2016) and ordered rules (Rasin *et al.* 2018), morphological segmentation (Goldsmith 2001, 2006), and inferring syntactic categories given known morphological patterns (Hu *et al.* 2005).

Context-free grammars

2.1

To demonstrate this idea in action, we will use the formalism of context-free grammars (CFGs), also called phrase-structure grammars (Chomsky 1956). CFGs were developed for describing syntactic structure in natural language and serve as the starting point of Chomsky’s (1965) Standard Theory. A context-free grammar is defined by specifying the following components:

- N , a finite set of *nonterminal symbols*. By convention, $S \in N$ is the *start symbol*;
- Σ , a finite set of *terminal symbols* disjoint from N ;
- R , a finite set of (*rewrite*) *rules*. Each member of R is a pair $\langle \alpha, \beta \rangle$ (usually written as $\alpha \rightarrow \beta$), where $\alpha \in N$ and β is a (potentially empty) string of terminal and nonterminal symbols.

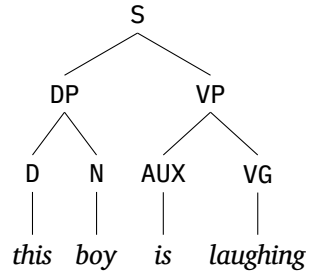
Rules are applied by replacing the nonterminal symbol on the left-hand side with the sequence on the right-hand side. The derivation begins with the start symbol and proceeds by applying rules until no nonterminal symbols are left in the string.

For a specific example, consider a CFG with $N = \{S, DP, VP, D, N, AUX, VG\}$, $\Sigma = \{this, boy, laughs, is, laughing\}$, and R as given in Figure 2a. CFGs are often represented simply as a list of rewrite rules, since N and Σ are recoverable from R . The phrase-structure tree, or

Figure 2:
A toy
context-free
grammar

- S → DP VP
- DP → D N
- D → *this*
- N → *boy*
- VP → *laughs*
- VP → AUX VG
- AUX → *is*
- VG → *laughing*

(a) Rules



(b) Parse tree of *this boy is laughing*

parse tree, associated with the derivation of the string *this boy is laughing*, is shown in Figure 2b. In a phrase-structure tree for a context-free derivation, each internal node corresponds to the left-hand side of a rule, and its children to symbols on the rule’s right-hand side.

Context-free grammars have been shown by Shieber (1985) to be insufficiently powerful to describe patterns found in natural language syntax. Nevertheless, they have useful connections to other grammar formalisms that will be discussed in Subsection 3.3.

2.2

Encoding FGs

Now let us consider a corpus of three strings over $\Sigma = \{this, boy, girl, laughs, jumps, and\}$:

this boy laughs;

this girl jumps;

this boy jumps and this girl laughs.

The three CFGs in Figure 3 all generate these strings but assign different phrase-structure trees to them (Figure 4). The first one (Figure 3a) is too permissive and *overgenerates* by producing every non-empty string in Σ^* , including those that are not grammatical sentences in English, such as **laughs jumps girl and this this*. In linguistic terms, Figure 3a assigns the same syntactic category to every word without regard to their distribution. The second grammar (Figure 3b) is too constraining and *overfits* the corpus: it generates the three sentences

	$S \rightarrow S_1 \text{ CONJ } S_2$	
	$S \rightarrow S_3$	
	$S \rightarrow S_4$	
	$S_1 \rightarrow DP_1 VP_2$	
	$S_2 \rightarrow DP_2 VP_1$	
	$S_3 \rightarrow DP_1 VP_1$	
	$S_4 \rightarrow DP_2 VP_2$	$S \rightarrow S \text{ CONJ } S$
$S \rightarrow X S$	$DP_1 \rightarrow D N_1$	$S \rightarrow DP VP$
$S \rightarrow X$	$DP_2 \rightarrow D N_2$	$DP \rightarrow D N$
$X \rightarrow \textit{this}$	$D \rightarrow \textit{this}$	$D \rightarrow \textit{this}$
$X \rightarrow \textit{boy}$	$N_1 \rightarrow \textit{boy}$	$N \rightarrow \textit{boy}$
$X \rightarrow \textit{girl}$	$N_2 \rightarrow \textit{girl}$	$N \rightarrow \textit{girl}$
$X \rightarrow \textit{laughs}$	$VP_1 \rightarrow \textit{laughs}$	$VP \rightarrow \textit{laughs}$
$X \rightarrow \textit{jumps}$	$VP_2 \rightarrow \textit{jumps}$	$VP \rightarrow \textit{jumps}$
$X \rightarrow \textit{and}$	$\text{CONJ} \rightarrow \textit{and}$	$\text{CONJ} \rightarrow \textit{and}$
(a) Overgenerating	(b) Overfitting	(c) Balanced

Figure 3:
Three
context-free
grammars

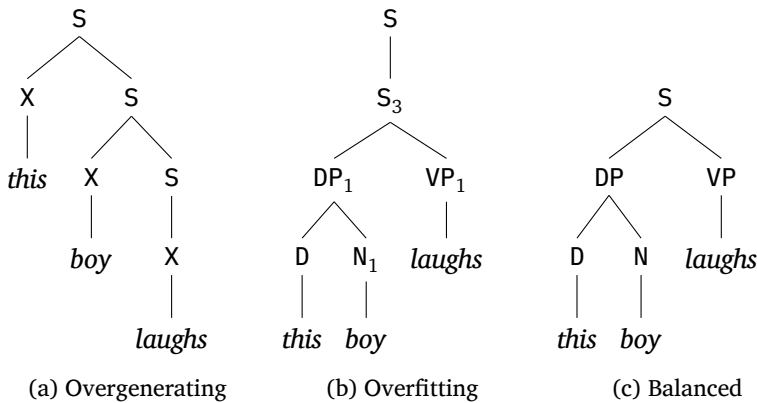


Figure 4:
Phrase-structure
trees for *this boy
laughs*

above and nothing else. Finally, Figure 3c strikes a balance by making a number of correct generalizations – for instance, that *boy* and *girl* have the same distribution and should be generated by the same nonterminal symbol. This grammar generates every sentence in the corpus, but also an infinite set of grammatical sentences absent from the corpus such as *this boy laughs and this girl jumps and this girl laughs*.

We will now adopt a straightforward encoding scheme and notation after Katzir (2014) and Rasin and Katzir (2019) to see how this intuition translates into MDL values. The first step is to convert each nonterminal in N and each terminal in Σ , along with an additional *delimiter* symbol, $\#$, into a binary string. Then the number of bits needed to represent each symbol is :

$$\lceil \log_2(|N| + |\Sigma| + 1) \rceil,$$

where $\lceil \cdot \rceil$ indicates rounding up to the nearest integer. For simplicity, this encoding scheme assigns binary strings of equal length to all symbols; see Section 6 for discussion of alternatives. It takes four bits to encode a symbol in Figure 3a or 3c, while the symbols of 3b require five bits each (Figure 5).

We can now use these binary representations to encode each grammar. Since context-free rewrite rules follow a very specific format (one nonterminal symbol on the left-hand side, a sequence of terminal and nonterminal symbols on the right-hand side), a grammar can be unambiguously represented by concatenating all symbols in each rule and concatenating all rules together, separated by delimiters, as shown in Figure 6.

This step converts a grammar into a single binary string. Formalizing, the length of this string equals

$$\sum_{\langle \alpha, \beta \rangle \in R} (|\alpha| + |\beta| + 1) \times \lceil \log_2(|N| + |\Sigma| + 1) \rceil$$

and represents the size of the entire grammar in bits.

2.3

Encoding corpora

Our next step is to encode the data, which is done by using phrase-structure trees of sentences in the corpus. We start at the root (labeled with the start symbol, S) and traverse the tree in preorder – i.e. read the current node, then recursively traverse its children in the same

	#	00000			
	S	00001			
	S ₁	00010			
	S ₂	00011			
	S ₃	00100			
	S ₄	00101			
	DP ₁	00110			
	DP ₂	00111	#	0000	
	D	01000	S	0001	
	N ₁	01001	DP	0010	
	N ₂	01010	D	0011	
#	0000	VP ₁	01011	N	0100
S	0001	VP ₂	01100	VP	0101
X	0010	CONJ	01101	CONJ	0110
<i>this</i>	0011	<i>this</i>	01110	<i>this</i>	0111
<i>boy</i>	0100	<i>boy</i>	01111	<i>boy</i>	1000
<i>girl</i>	0101	<i>girl</i>	10000	<i>girl</i>	1001
<i>laughs</i>	0110	<i>laughs</i>	10001	<i>laughs</i>	1010
<i>jumps</i>	0111	<i>jumps</i>	10010	<i>jumps</i>	1011
<i>and</i>	1000	<i>and</i>	10011	<i>and</i>	1100

(a) Overgenerating

(b) Overfitting

(c) Balanced

Figure 5:
Encoding tables
for symbols

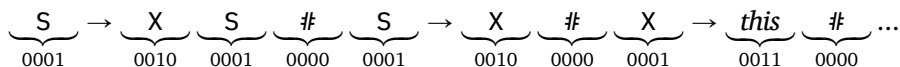


Figure 6: Encoding of the overgenerating grammar (Figure 3a)

way, from left to right. At each internal node, the number of possible choices equals the number of different rules whose left-hand side corresponds to the node's label. Formally, given the left-hand side α , the cost of encoding a rule in bits is: $\lceil \log_2(|\{\beta : \langle \alpha, \beta \rangle \in R\}|) \rceil$.

Using the overfitting grammar (Figure 3b) as an example, the cost of using the rule $S \rightarrow S_3$ given the left-hand side S is $\lceil \log_2 3 \rceil = 2$ bits, because there are 3 different rules whose left-hand side is S. If there is

		$S \rightarrow S_1 \text{ CONJ } S_2$	00		
		$S \rightarrow S_3$	01		
		$S \rightarrow S_4$	10		
		$S_1 \rightarrow DP_1 VP_2$	ϵ		
		$S_2 \rightarrow DP_2 VP_1$	ϵ		
		$S_3 \rightarrow DP_1 VP_1$	ϵ		
		$S_4 \rightarrow DP_2 VP_2$	ϵ	$S \rightarrow S \text{ CONJ } S$	0
$S \rightarrow X S$	0	$DP_1 \rightarrow D N_1$	ϵ	$S \rightarrow DP VP$	1
$S \rightarrow X$	1	$DP_2 \rightarrow D N_2$	ϵ	$DP \rightarrow D N$	ϵ
$X \rightarrow \textit{this}$	000	$D \rightarrow \textit{this}$	ϵ	$D \rightarrow \textit{this}$	ϵ
$X \rightarrow \textit{boy}$	001	$N_1 \rightarrow \textit{boy}$	ϵ	$N \rightarrow \textit{boy}$	0
$X \rightarrow \textit{girl}$	010	$N_2 \rightarrow \textit{girl}$	ϵ	$N \rightarrow \textit{girl}$	1
$X \rightarrow \textit{laughs}$	011	$VP_1 \rightarrow \textit{laughs}$	ϵ	$VP \rightarrow \textit{laughs}$	0
$X \rightarrow \textit{jumps}$	100	$VP_2 \rightarrow \textit{jumps}$	ϵ	$VP \rightarrow \textit{jumps}$	1
$X \rightarrow \textit{and}$	101	$\text{CONJ} \rightarrow \textit{and}$	ϵ	$\text{CONJ} \rightarrow \textit{and}$	ϵ
(a) Overgenerating		(b) Overfitting		(c) Balanced	

Figure 7: Encoding tables for rules

Table 1:
Encoding costs
for Figure 3a–3c
(bits)

	Grammar	Corpus	MDL
Overgenerating (Figure 3a)	100	52	152
Overfitting (Figure 3b)	265	6	271
Balanced (Figure 3c)	124	13	137

only one possible right-hand side, as with the rule $S_3 \rightarrow DP_1 VP_1$, the cost is 0 bits because there is no choice to make, and the corresponding encoding is ϵ , the empty string.

In this way, we can now give binary string representations to all rules, as shown in Figure 7. To encode a tree, we concatenate all rule encodings in the order in which the nodes are traversed (Figure 8).

This explicit encoding scheme highlights the differences in how each grammar describes the data. Overall costs for the three grammars and data are given in Table 1. The overgenerating grammar (Figure 3a) is very short but requires a lengthy encoding of the corpus.

$$\underbrace{S \rightarrow X S}_0 \quad \underbrace{X \rightarrow \textit{this}}_{000} \quad \underbrace{S \rightarrow X S}_0 \quad \underbrace{X \rightarrow \textit{boy}}_{001} \quad \underbrace{S \rightarrow X}_1 \quad \underbrace{X \rightarrow \textit{laughs}}_{011}$$

(a) Overgenerating

$$\underbrace{S \rightarrow S_3}_{01} \quad \underbrace{S_3 \rightarrow DP_1 VP_1}_{\epsilon} \quad \underbrace{DP_1 \rightarrow D N_1}_{\epsilon} \quad \underbrace{D \rightarrow \textit{this}}_{\epsilon} \quad \underbrace{N_1 \rightarrow \textit{boy}}_{\epsilon} \quad \underbrace{VP_1 \rightarrow \textit{laughs}}_{\epsilon}$$

(b) Overfitting

$$\underbrace{S \rightarrow DP VP}_1 \quad \underbrace{DP \rightarrow D N}_{\epsilon} \quad \underbrace{D \rightarrow \textit{this}}_{\epsilon} \quad \underbrace{N \rightarrow \textit{boy}}_0 \quad \underbrace{VP \rightarrow \textit{laughs}}_0$$

(c) Balanced

Figure 8:
Encoding of
this boy laughs

The overfitting grammar (Figure 3b) makes describing the corpus extremely easy at the cost of a long encoding of the grammar itself.

The sum of the grammar and corpus encoding favors the balanced grammar (Figure 3c) – which aligns with a linguistic intuition of which of the three grammars is best.⁵

MINIMALIST GRAMMARS

3

Lexical items, Merge, and Move

3.1

Minimalist grammars (MGs, Stabler 1997) provide a formal implementation of Minimalist syntax (Chomsky 1995, 2000), which is used throughout the paper. In order to keep the paper fully self-contained, this section introduces the MG formalism and provides examples of derivations.

⁵ An editor has pointed out that the following “extremely overfitting” grammar would outperform the balanced grammar given the corpus discussed above:

$S \rightarrow \textit{this boy laughs}$

$S \rightarrow \textit{this girl jumps}$

$S \rightarrow \textit{this boy jumps and this girl laughs}$

This grammar introduces no nonterminal symbols other than S, which works well for the three-sentence corpus. However, we can easily construct an example over

An MG specifies a finite set of lexical items and encodes their selectional properties in the form of *syntactic features*. A feature of the form x corresponds to a syntactic *category*, whereas $=x$, $=>x$, and $x=$ are selecting features which indicate that an expression is looking to merge (on the right, on the right with head movement, or on the left, respectively⁶) with something of that category. Similarly, $-x$ indicates

the same Σ that would make better use of additional nonterminals and show extreme overfitting underperform on a slightly larger dataset.

Let us add to the original corpus a sentence containing $n + 1$ clauses (for some n) of the form: *this boy laughs and this girl jumps ... and this girl jumps*. On

the grammar side, the overgenerating and balanced grammar can already generate it. The overfitting grammar needs to add two new rules: $S \rightarrow S_5$ and $S_5 \rightarrow S_3 \text{ CONJ } S_4 \dots \text{ CONJ } S_4$. The extremely overfitting grammar needs one rule, $S \rightarrow \text{this boy laughs } \underbrace{\text{and this girl jumps } \dots \text{ and this girl jumps}}_{n \text{ times}}$, costing three

bits per symbol to encode. On the corpus side, the overgenerating grammar would pay one bit per word in the sentence to choose between $S \rightarrow S X$ and $S \rightarrow X$ and three bits per word to pick the terminal. For the balanced grammar, the additional cost is n instances of $S \rightarrow S \text{ CONJ } S$, $n + 1$ instances of $S \rightarrow \text{DP VP}$, and two more bits per clause to pick the noun and the verb. Both the overfitting and the extremely overfitting grammar would see a flat 2-bit increase.

	Grammar cost increase	Corpus cost increase
Overgenerating	0	$(3 + 4n) + 3 \times (3 + 4n)$
Balanced	0	$n + (n + 1) + 2 \times (n + 1)$
Overfitting	$5 \times 3 + 5 \times (3 + 2n)$	2
Extremely overfitting	$3 \times (5 + 4n)$	2

It is easy to see that the balanced approach and even the overfitting one outperform extreme overfitting at higher values of n . While not very natural (as the number of distinct words is limited to keep it simple), this example shows how the initial investment of setting up syntactic structure (as additional nonterminal symbols and rules) takes more than a toy corpus to pay off.

⁶The choice to distinguish between left and right selection puts linear order under lexical control. One alternative, commonly adopted in the literature on MGs, is to have the first dependent of a head merge on the right, and all subsequent dependents on the left – a version of the Linear Correspondence Axiom (Kayne 1994).

the requirement to move, and $+x$ and $\star x$ mean that the expression attracts a sub-expression with that feature into its specifier position (overtly or covertly).

In order to define an MG, one has to specify the following:

- *Base*, a finite set of syntactic *categories*. The set *Syn* of syntactic features is defined as the union of *Base* and the following sets:

$$\begin{aligned}
 Sel &= \{=x : x \in Base\} \cup && \text{(right selectors)} \\
 &\{>x : x \in Base\} \cup && \text{(morphological selectors)} \\
 &\{x= : x \in Base\} && \text{(left selectors)} \\
 Lic &= \{+x : x \in Base\} \cup && \text{(overt licensors)} \\
 &\{\star x : x \in Base\} && \text{(covert licensors)} \\
 Lee &= \{-x : x \in Base\} && \text{(licensees)}
 \end{aligned}$$

Each syntactic feature is then characterized by its *name* (drawn from *Base*) and *type* (category, right/morphological/left selector, overt/covert licensor, or licensee). Selectors and licensors together are called *attractors*, and categories and licensees are called *attractees*;

- Σ , a finite alphabet of phonological segments;
- *Lex*, a lexicon, or finite set of *lexical items*. Each lexical item (LI) is a pair $\langle s, \delta \rangle$ (written as $s :: \delta$), where $s \in \Sigma^*$ is a (phonological) *string component* and $\delta \in Syn^*$ is a list of syntactic features, or *feature bundle*. In cases that are not ambiguous, we will sometimes refer to specific lexical items by their string components.

MGs are commonly defined by simply stating a lexicon, which also implicitly fixes a set of categories and an alphabet of segments. Because of this, and for the sake of convenience, we will use the terms “grammar” and “lexicon” interchangeably when referring to MGs. An example grammar of five lexical items is given in Figure 9.⁷

Syntactic *expressions* generated by an MG are binary trees whose terminal nodes are labeled with LIs (which themselves are referred to

⁷In this example, all complements are merged on the right. The subject DP then moves to the position to the left of the finite verb.

Figure 9:
A toy MG

this :: =n d -k
boy :: n
is :: =g +k t
laugh :: =d v
-ing :: =>v g
-s :: =>v +k t

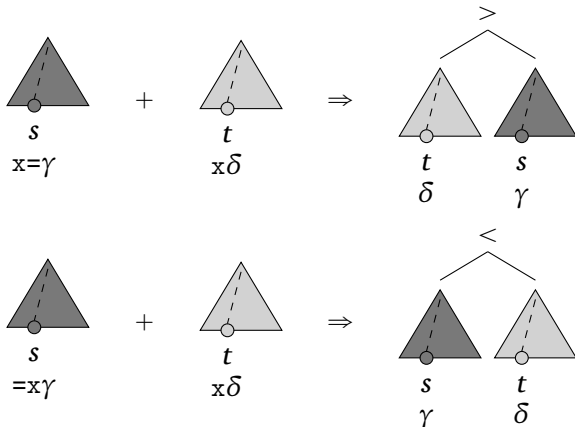
as *atomic expressions*). The first feature of each LI is *syntactically active*, i.e. accessible to structure building operations. These operations, **merge** and **move**, consume matching attractors and attractees to generate complex expressions from *Lex*.

Following Stabler (2001), head movement is implemented as a subtype of **merge**, driven by features of the form =>x , which we will call *morphological selectors*. This version of head movement is defined in terms of head-complement relations, which means that this type is restricted to the first feature in the bundle. This addition allows minimalist lexica to reflect structure within complex words.⁸ We will refer to lexical items bearing these selector features as *affixes* and write their string components starting with a hyphen, following a common notational convention.

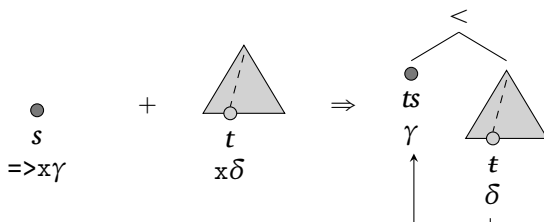
The set of expressions *Exp* is defined as the closure of *Lex* under **merge** and **move**.

⁸Regarding the issue of complex words, multiple options have been explored in the literature. Head movement creates a chain of heads that is pronounced in the highest head position. 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. Unification of head movement and lowering is one of the defining features of Brody's (2000) Mirror Theory. 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*. Stabler (2001) incorporates both head movement and lowering into MGs as subtypes of selector features. 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 by Kobele (to appear). In this paper, we consider all complex words to be formed by head movement. This decision is explicitly treated as a simplifying assumption.

- **merge** : $(Exp \times Exp) \rightarrow Exp$ is a binary function that targets selectors and categories and combines two syntactic expressions into a new one. The dependent is merged on the left if the selector is of the form $x=$, and on the right if it is of the form $=x$:



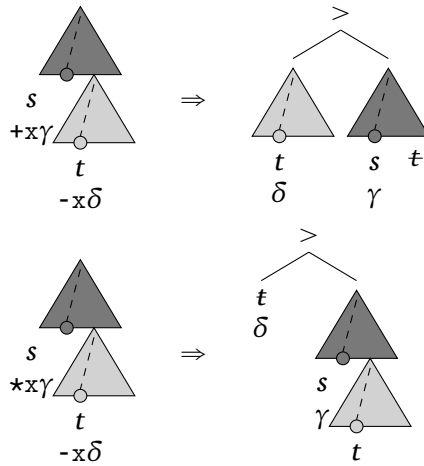
- **merge with head movement** is triggered by selectors of the form $=>x$. It proceeds as right merge and concatenates the string component of the head of the complement with that of the resulting expression:⁹



- **move** : $Exp \times Exp$ is a relation that matches a licenser with a licensee within the same expression. Overt licensers ($+x$) cause the moving subtree to become a (left) sister of the head, leaving behind an empty node without a string component or syntactic features. Covert **move** ($*x$) leaves the string component behind:¹⁰

⁹We indicate a moved string t as \acute{t} . This is a notational convenience; formally, the empty node contains ϵ , the empty string.

¹⁰This version of covert movement, which displaces syntactic features but leaves the string component in its base position, is in line with Stabler 1997. It fixes the position of a sub-expression once it has been covertly moved, rendering its string component inaccessible to future instances of (overt) move. Though



While there are many ways to limit the number of features which may be syntactically active at any given time, a simple one with desirable computational properties stipulates that only one feature of each name may be the first feature of any feature bundle in an expression. In particular, this means that the number of movable subtrees in any expression is limited by the size of *Base*. This restriction is known as the Shortest Move Constraint, or SMC. With the SMC in place, *move* becomes a function.

A single lexical item (atomic expression) is considered its own *head*. For complex structures formed by *merge* or *move*, the expression with the attractor becomes the head of the new expression; and the one with the attractee becomes its *dependent*. We label the parent node with < if the head is on the left or > if the head is on the right. The dependent introduced by the first attractor of an LI is its *complement*, and all subsequent dependents are *specifiers*. Matched features are *checked*, or deleted, making the next feature in the bundle accessible for syntactic operations. Checked features are no longer visible to syntax. We will sometimes keep them in representations for clarity, in which case they will be marked as \boxed{x} .

An expression with no unchecked features except for some category *x* on its head is called a *complete expression* of that category.

restricted, this implementation has been used in previous work on MGs (see e.g. Torr and Stabler 2016) and is sufficient for our purposes.

We will be primarily concerned with complete expressions of category t (for Tense) or c (for Complementizer) and their string yields (*sentences*).

The lexicon in Figure 9 generates, among others, the five expressions in Figure 10. In Figure 10a, *merge* applies to *this* and *boy*, whose

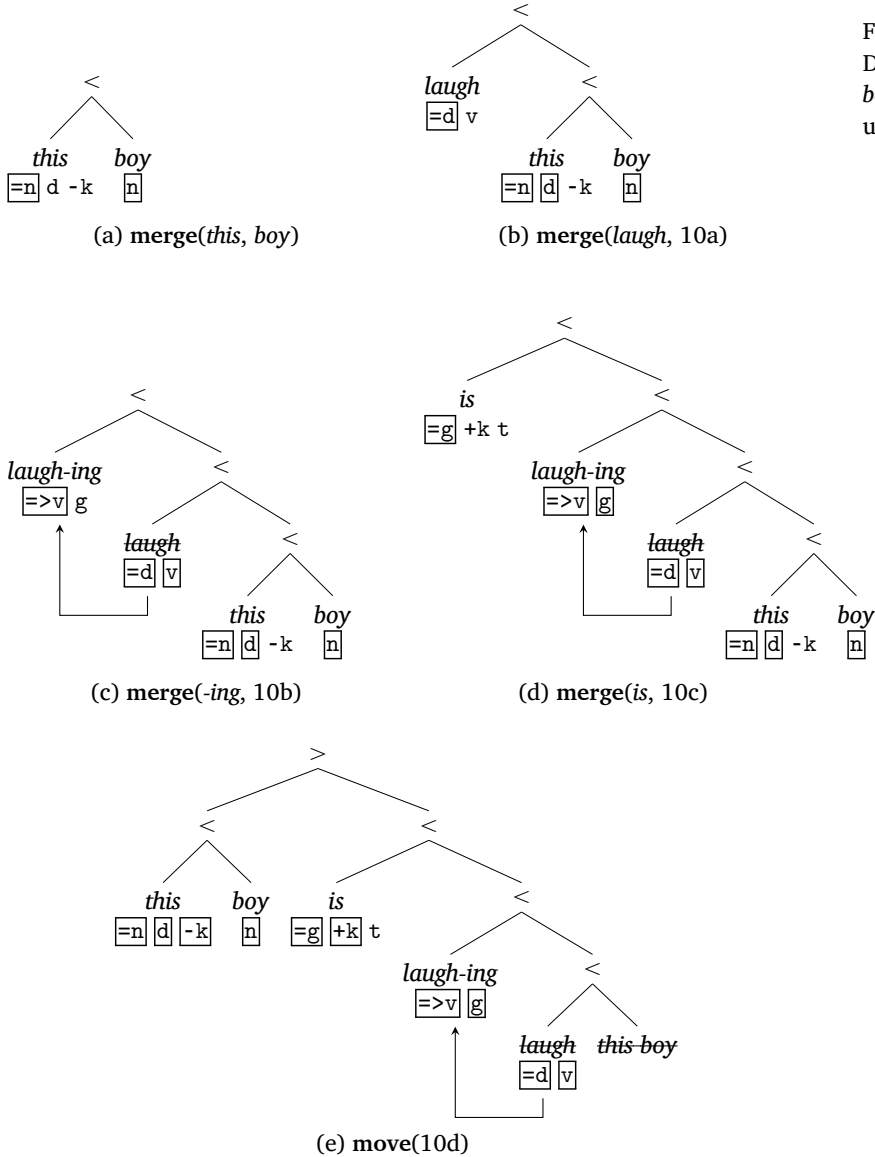


Figure 10:
Derivation of *this
boy is laugh-ing*
using Figure 9

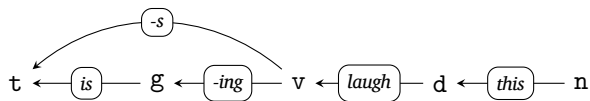
feature bundles start with the matching features =n and n, respectively. Both =n and n are deleted. In Figure 10b, **merge** once again targets two expressions: *laughing*'s feature bundle starts with =d, and Figure 10a has d as its first feature. Next, we **merge in** *-ing*. Its selector feature, =>v, triggers head movement, concatenating *laugh* and *-ing* together (Figure 10c). Another **merge** step (Figure 10d) checks the =g and g features, combining *is* with Figure 10c. In Figure 10e, the matching features are +k on *is* and -k on *this*. The DP is **moved** into the specifier position of *is*, which becomes the head of the new expression. This is a complete expression of category t, whose string yield is *this boy is laugh-ing*.

3.2

Grammar graphs

When it comes to visualizing an entire grammar, the default option is to list all lexical items, as in Figure 9. As mentioned before, such a list contains all information required to define an MG. However, it does not provide a good overview of expressions generated by the grammar in question. While it works for very small toy examples, larger grammars with dozens or hundreds of LIs can become difficult to read quickly. A convenient alternative for showing the head-complement relations within a set of lexical items is a directed multigraph whose vertices correspond to category features, and edges to lexical items. To better understand, consider Figure 11 which illustrates this representation using the same data as Figure 9.

Figure 11:
Head-complement relations
within Figure 9



This graph does not reflect all relations in the lexicon, since it ignores any **move** relations as well as any specifiers formed by **merge**. Lexical items without any selectors (such as *boy* :: n) don't contribute an edge to the graph. Instead, it focuses on a subset of relations which are relevant for morphologically complex words. 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 an expression of category v can be selected either by $-s :: =>v +k t$ or by $-ing :: =>v g$, in the latter case producing a valid complement for $is :: =g +k t$.

Relation to CFGs

3.3

By definition, the two structure-building operations of MGs – **merge** and **move** – can only target subtrees whose heads bear an unchecked syntactic feature. Therefore, much of the derived structure is *syntactically inert*: once all features of a lexical item have been deleted, its position in the structure is fixed. The only elements that matter for syntax are those still capable of rearranging with respect to each other – namely, the head of the entire expression (via head movement) and any *movers*, or subtrees headed by lexical items with an unchecked licensee feature. With the SMC in place, the number of such subtrees in any given expression is finite, limited by the number of distinct licensee features in the grammar. Thus, a derived tree can be flattened into a much more compact structure containing all information relevant for **merge** and **move** – a sequence of strings annotated with unchecked features.

This insight gives rise to the so-called *chain notation* for MGs (Stabler 2001; Stabler and Keenan 2003). In short, each expression sans movers is represented as an *initial chain* – a triple of strings corresponding to the head and material to its left and right, annotated with features of the head. Movers within the tree are represented by separate *non-initial chains*, the number of which cannot be greater than the size of *Base* (see Figure 12).

$$\underbrace{(left, head, right) : features}_{\text{Initial chain}}, \underbrace{mover_1 : licensees, mover_2 : licensees, \dots}_{\text{Non-initial chains}}$$

Figure 12: Schematic representation of a chain-based expression

Lexical items consist of only an initial chain, and their first and last components are empty strings, as shown in Figure 13.

The structure-building operations are redefined in terms of string tuples. Informally, the outcome of **merge** depends on whether the dependent has reached its final position in the structure or is going to

Figure 13:
Chain-based counterpart of Figure 9

$$\begin{aligned} \langle \epsilon, \textit{this}, \epsilon \rangle &:: =n \textit{d} -k \\ \langle \epsilon, \textit{boy}, \epsilon \rangle &:: n \\ \langle \epsilon, \textit{is}, \epsilon \rangle &:: =g +k \textit{t} \\ \langle \epsilon, \textit{laugh}, \epsilon \rangle &:: =d \textit{v} \\ \langle \epsilon, \textit{-ing}, \epsilon \rangle &:: =>\textit{v} \textit{g} \\ \langle \epsilon, \textit{-s}, \epsilon \rangle &:: =>\textit{v} +k \textit{t} \end{aligned}$$

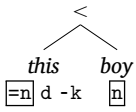
move later in the derivation. In the former case, its initial chain is concatenated together and attached to the leftmost (for left **merge**) or rightmost (for right **merge**) component of the initial chain. In the latter case, the dependent forms a non-initial chain ready to be targeted by **move**. Similarly, **move** comes in multiple varieties depending on whether the moving subtree has reached its surface position. A complete expression of category x consists of just an initial chain annotated with only the feature x .

The derivation of *this boy is laugh-ing*, shown before in Figure 10, is repeated in Figure 14, with each derivation step given as a derived tree and in chain notation side by side. In Figure 14a, *this* and *boy* are **merged**, and the string component of the latter is concatenated into the third component of the initial chain. Next, *laugh* is **merged** with the resulting structure (Figure 14b). Since the dependent still carries a license feature ($-k$), it forms a non-initial chain *this boy* annotated with $-k$. The next two steps continue building up the initial chain, leaving the single non-initial chain unaffected. Finally, Figure 14e **moves** *this boy* into the first component of the initial chain, arriving at a complete expression of category t .

Because chain notation is so compact, all intermediate steps in a derivation can be visualized as a single *derivation tree* by labeling each internal node with the chain-based expression corresponding to the step in question, as shown in Figure 15. Each internal node corresponds to a step in the derivation, an instance of **merge** or **move**, and the order of its children reflects their role in that step: the head precedes its dependent regardless of their relative order in the derived structure.

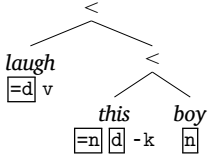
Derivation trees don't reflect displacement of leaves caused by **move** in the way derived trees do. For any MG, its derivation trees are

Evaluating syntactic proposals using MGs and MDL



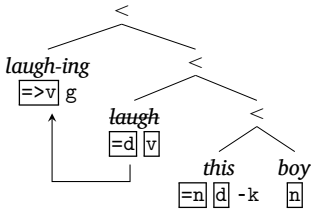
$\langle \epsilon, \text{this}, \text{boy} \rangle : d -k$

(a) merge(*this*, *boy*)



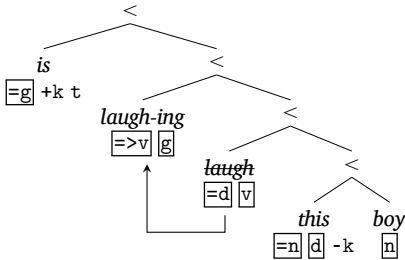
$\langle \epsilon, \text{laugh}, \epsilon \rangle : v, \text{this boy} : -k$

(b) merge(*laugh*, 14a)



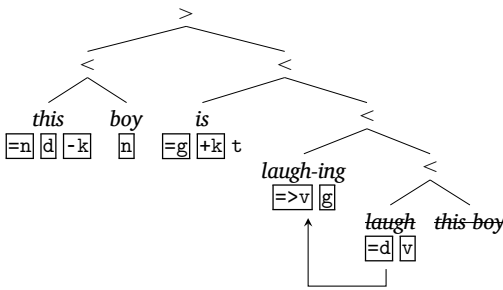
$\langle \epsilon, \text{laugh-ing}, \epsilon \rangle : g, \text{this boy} : -k$

(c) merge(-ing, 14b)



$\langle \epsilon, \text{is}, \text{laugh-ing} \rangle : +k t, \text{this boy} : -k$

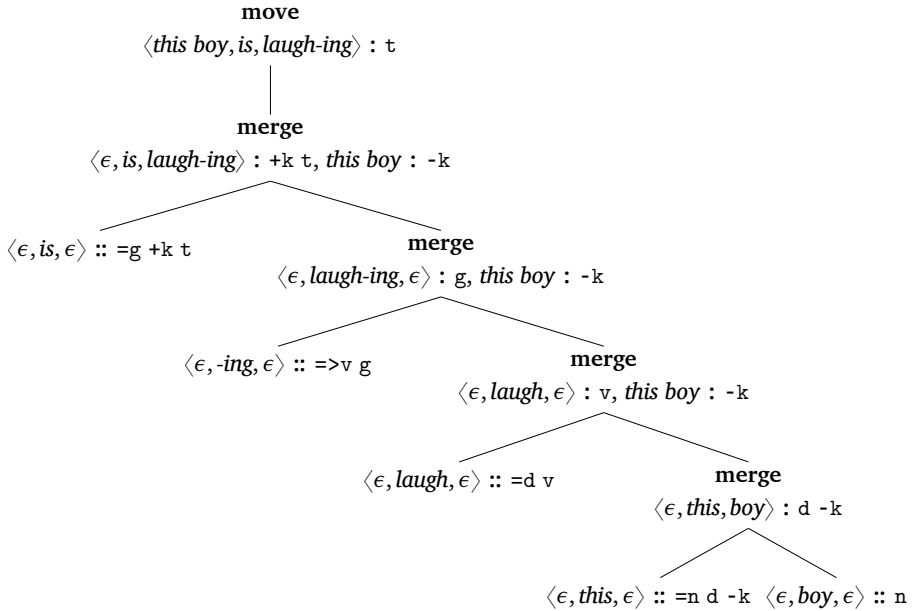
(d) merge(*is*, 14c)



$\langle \text{this boy}, \text{is}, \text{laugh-ing} \rangle : t$

(e) move(14d)

Figure 14: Derivation steps of *this boy is laugh-ing* as chain-based expressions

Figure 15: Chain-based derivation tree of *this boy is laugh-ing*

parse trees of a CFG; a clear presentation of this result is given in (Hale and Stabler 2005). Intuitively, constructing this CFG can be thought of as pre-computing all possible derivation steps that can be performed by the MG. The central concept here is that of a *feature configuration*, which is obtained from a chain-based expression by omitting string components;¹¹ the SMC guarantees that the number of such configurations is finite. The set of feature configurations is obtained as the closure of the lexicon under **merge** and **move**. Informally, the conversion process is as follows:

- Each feature configuration (written in round brackets) becomes a nonterminal symbol;
- For each feature configuration formed by **merge** or **move**, there is a rule rewriting it as the operation's argument or arguments;
- For each LI, there is a rule rewriting its feature configuration as its string component;

¹¹For covert movement, feature configurations should also indicate the non-initial chains whose string components have been left behind.

- An additional rule rewrites the start symbol S as (t) or (c) .¹²

Derivation is then viewed as proceeding in the top-down manner of CFGs (starting with t and rewriting until lexical items in the leaves are reached), rather than the bottom-up manner characteristic of MGs. The CFG obtained from Figure 9 is shown in Figure 16.

$$\begin{array}{l}
 S \rightarrow (t) \\
 (t) \rightarrow (+k \ t, \ -k) \quad (=n \ d \ -k) \rightarrow \textit{this} \\
 (+k \ t, \ -k) \rightarrow (=g \ +k \ t) \ (g, \ -k) \quad (n) \rightarrow \textit{boy} \\
 (+k \ t, \ -k) \rightarrow (=>v \ +k \ t) \ (v, \ -k) \quad (=g \ +k \ t) \rightarrow \textit{is} \\
 (g, \ -k) \rightarrow (=>v \ g) \ (v, \ -k) \quad (=d \ v) \rightarrow \textit{laugh} \\
 (v, \ -k) \rightarrow (=d \ v) \ (d \ -k) \quad (=>v \ g) \rightarrow \textit{-ing} \\
 (d \ -k) \rightarrow (=n \ d \ -k) \ (n) \quad (=>v \ +k \ t) \rightarrow \textit{-s}
 \end{array}$$

Figure 16:
CFG counterpart
of Figure 9

ENCODING MINIMALIST GRAMMARS

4

With these definitions in place, we will now discuss how the approach of Section 2 can be adapted to implement an MDL-based metric for MGs. Consider the following four sentences:

Mary laughs;
Mary laughed;
Mary jumps;
Mary jumped.

¹²The method given in Hale and Stabler 2005 is itself an adaptation of Michaelis 1998, which shows how to convert an MG into an equivalent multiple context-free grammar (MCFG) generating the same language of sentences – yields of derived trees. MCFGs are a generalization of CFGs which operates on tuples instead of strings. Converting an MG into an equivalent MCFG is similar to the CFG construction, with a few differences. First, terminal rules rewrite feature bundles as triples of strings, corresponding to initial chains. Second, each non-terminal rule comes with a map describing how components of the argument tuples are rearranged and/or concatenated, in a way closely following chain-based **merge** and **move**.

There are multiple (in fact, infinitely many) ways to construct a minimalist grammar accounting for this small corpus. Three of them are given in Figure 17.

Figure 17:
Three minimalist
grammars

<i>Mary</i> :: d -k	<i>Mary</i> :: d -k	<i>Mary</i> :: x -k
<i>laughs</i> :: =d +k t	<i>laugh</i> :: =d v	<i>laugh</i> :: =x x
<i>laughed</i> :: =d +k t	<i>jump</i> :: =d v	<i>jump</i> :: =x x
<i>jumps</i> :: =d +k t	<i>-s</i> :: =>v +k t	<i>-s</i> :: =>x +k t
<i>jumped</i> :: =d +k t	<i>-ed</i> :: =>v +k t	<i>-ed</i> :: =>x +k t
(a) Atomic verbs	(b) Complex verbs	(c) Overgenerating

The first two grammars, Figure 17a and 17b, generate the four sentences above and no others. While they are *weakly equivalent*, i.e. generate exactly the same set of strings, the structures they assign to these strings are different. In linguistic terms, the former treats each sentence as a single $\bar{t}P$ headed by an unsegmented verb. The latter reanalyzes each finite verb form as a complex head formed by head movement. The lexical verb directly selects its argument and forms a $\bar{v}P$, while the affix takes the $\bar{v}P$ as its complement and is responsible for the movement of the subject into its specifier position (Figure 18b). The third grammar, Figure 17c, is also capable of generating inflected verbs in two derivation steps (Figure 18c). However, it conflates the category feature of lexical verbs with that of DPs, producing ungrammatical strings like **Mary-ed* and **Mary laugh-s (jump)⁺* (Figure 19).

To further help visualize the differences between these grammars, their graph representations are given in Figure 20.

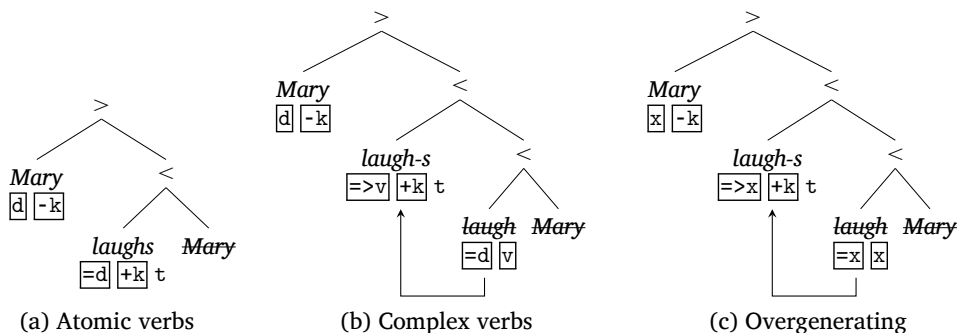


Figure 18: Structural differences

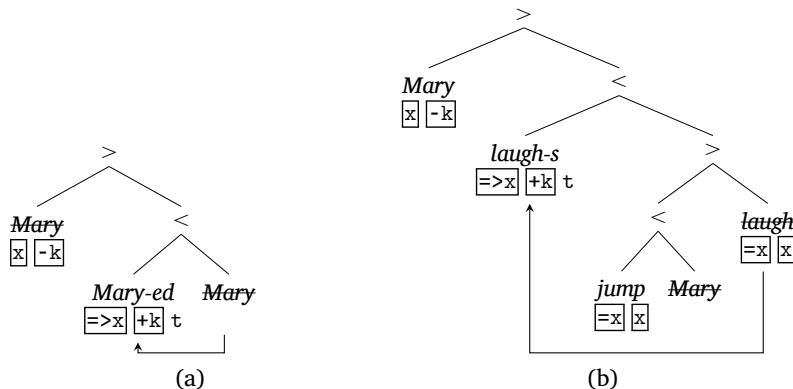


Figure 19:
Overgeneration
by Figure 17c

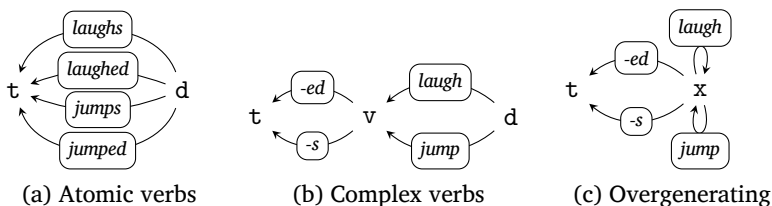


Figure 20:
Graph
representations
of the grammars
in Figure 17

For instance, *laughed* in the atomic-verb grammar corresponds to one of the edges from *d* to *t* in Figure 20a. Its counterpart in the grammar with complex verbs is a bimorphemic word, which translates into a pair of adjacent edges: *laugh* from *d* to *v* and *-ed* from *v* to *t* (Figure 20b). In the overgenerating grammar, lexical verbs correspond to loops (Figure 20c).

Intuitively, complex verbs are an improvement over atomic verbs. By recognizing internal structure within verbs, it captures the similarities within verbal paradigms (*laughs, laughed* vs. *jumps, jumped*) and across paradigms (*laughs, jumps* vs. *laughed, jumped*). On the other hand, atomic verbs miss all these generalizations. For each new verbal paradigm encountered in the corpus (e.g. *walks, walked*), we would need to add two new lexical items to Figure 17a, but only one to Figure 17b. Finally, Figure 17c is a subpar choice: it shares the desirable generalizations of Figure 17b but also conflates a crucial distinction between two syntactic categories, leading to overgeneration.

What quantitative data can be used to back up this intuition? We can define an encoding scheme for MGs closely mirroring the one for context-free rules from Section 2. Since we are interested in the length

of the encoding rather than the binary string itself (Grünwald 2007), we no longer round up to the nearest integer. Let $Types = \{category, right\ selector, left\ selector, morphological\ selector, overt\ licenser, covert\ licenser, licensee\}$ denote the set of syntactic feature types, and let Σ be the set of English letters. For simplicity, as with context-free rules, we treat each LI as a sequence of symbols from the same encoding table. Then the size of a minimalist lexicon Lex over a set of categories $Base$ is given by

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

Assuming that both Σ and $Types$ are fixed (with $|Types| = 7$ and $|\Sigma| = 26$, without distinguishing between uppercase and lowercase letters), this is a function of the number of LIs and the following three metrics:

- $|Base|$, the number of unique category features in Lex ;
- $\sum_{syn} = \sum_{s :: \delta \in Lex} (|\delta|)$, the total count of syntactic features in Lex ;
- $\sum_{phon} = \sum_{s :: \delta \in Lex} (|s|)$, the total length of all string components in Lex .

Regardless of the specific encoding scheme,¹³ all three values above contribute to the size difference between grammars. Table 2 summarizes the differences between the grammars with respect to individual metrics, as well as grammar size.

Table 2:
Grammar metrics

	$ Base $	\sum_{syn}	\sum_{phon}	Grammar (bits)
Atomic verbs (Figure 17a)	3	14	28	317.78
Complex verbs (Figure 17b)	4	12	16	236.16
Overgenerating (Figure 17c)	3	12	16	234.43

All three grammars have the same number of lexical items. However, splitting verbs into roots and affixes in Figure 17b comes at the

¹³The solution used here serves to keep the example straightforward. The choice of an encoding scheme is a meaningful decision that can lead to different grammars being optimal for the same corpus; see also discussion in Section 6.

cost of an extra category feature. This pays off by eliminating redundant strings, which almost halves \sum_{phon} . Moreover, four instances of $+k$ are collapsed into two, yielding a small reduction of \sum_{syn} . The differences would be much more noticeable with larger datasets, especially with respect to open-class words, since adding a new verb to Figure 17a would have a higher cost (in both syntactic features and string components) compared to Figure 17b.

It is also easy to see how a complexity measure based solely on grammar encoding would fail to penalize overgeneration. It would incorrectly favor Figure 17c over Figure 17b, given that it achieves the same reduction of \sum_{phon} and \sum_{syn} without increasing $|Base|$. Similar to the results observed with CFGs, the MDL component expected to rule out the overgenerating grammar is the corpus size given the grammar. In order to calculate it, for each MG we construct a CFG generating its derivation trees, as we did in Subsection 3.3, and then reuse the encoding scheme from Section 2. The CFGs are given in Figure 21. Parse trees for *Mary laughs* as well as Figure 21c’s ungrammatical structures are shown in Figure 22 and Figure 23 respectively .

The cost of encoding the corpus given Figure 21a is straightforward to calculate: there is only one choice with four options to be made in the derivation, namely rewriting $(=d +k t)$ as *laughs*, *laughed*, *jumps*, or *jumped*. In Figure 21b this corresponds to two binary choices: rewriting $(=d v)$ as *laugh* or *jump*, and $(=>x +k t)$ as *-s* or *-ed*. Both cost 2 bits per sentence. The third grammar (Figure 21c), however, has two options for rewriting $(+k t, -k)$ and two ways to expand $(x, -k)$. These are the choices that make possible the ungrammatical strings in Figure 23, but they also drive up the cost of encoding each grammatical sentence to 4 bits. This is summarized in Table 3.

	Grammar	Corpus	MDL
Atomic verbs (Figure 17a)	317.78	8	325.78
Complex verbs (Figure 17b)	236.16	8	244.16
Overgenerating (Figure 17c)	234.43	16	250.43

Table 3:
Encoding costs (bits)

Once we take the length of corpus encoding into account, the overgenerating grammar is outperformed by the intuitively superior grammar with complex verbs.

<p> $S \rightarrow (t)$ $(t) \rightarrow (+k t, -k)$ $(+k t, -k) \rightarrow (=d +k t) (d -k)$ $(d -k) \rightarrow \textit{Mary}$ $(=d +k t) \rightarrow \textit{laughs}$ $(=d +k t) \rightarrow \textit{laughed}$ $(=d +k t) \rightarrow \textit{jumps}$ $(=d +k t) \rightarrow \textit{jumped}$ </p>	<p> $S \rightarrow (t)$ $(t) \rightarrow (+k t, -k)$ $(+k t, -k) \rightarrow (=>v +k t) (v, -k)$ $(v, -k) \rightarrow (=d v) (d -k)$ $(d -k) \rightarrow \textit{Mary}$ $(=d v) \rightarrow \textit{laugh}$ $(=d v) \rightarrow \textit{jump}$ $(=>v +k t) \rightarrow -s$ $(=>v +k t) \rightarrow -ed$ </p>
<p>(a) Atomic verbs</p>	<p>(b) Complex verbs</p>
	<p> $S \rightarrow (t)$ $(t) \rightarrow (+k t, -k)$ $(+k t, -k) \rightarrow (=>x +k t) (x, -k)$ $(+k t, -k) \rightarrow (=>x +k t) (x -k)$ $(x, -k) \rightarrow (=x x) (x -k)$ $(x, -k) \rightarrow (=x x) (x, -k)$ $(x -k) \rightarrow \textit{Mary}$ $(=x x) \rightarrow \textit{laugh}$ $(=x x) \rightarrow \textit{jump}$ $(=>x +k t) \rightarrow -s$ $(=>x +k t) \rightarrow -ed$ </p>
	<p>(c) Overgenerating</p>

Figure 21: CFG counterparts of Figure 17

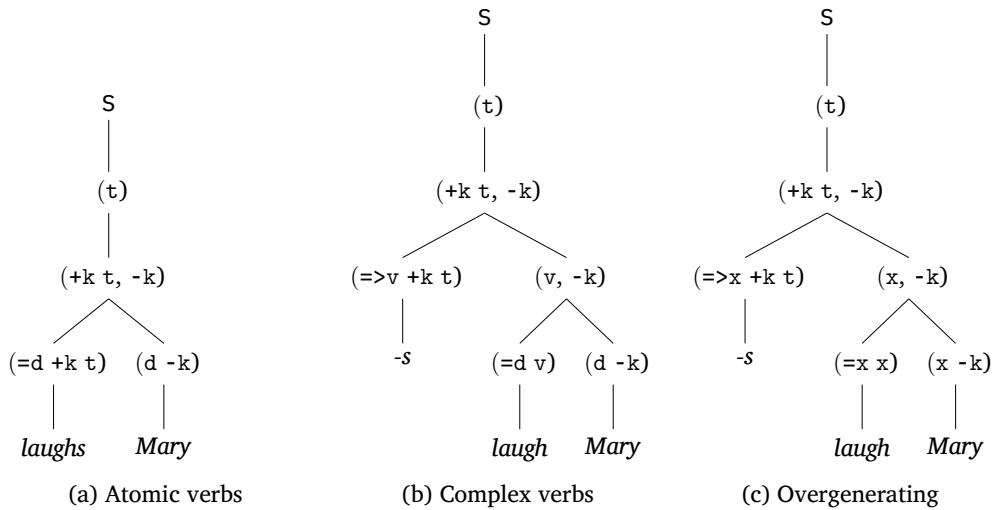


Figure 22: CFG parse trees: structural differences

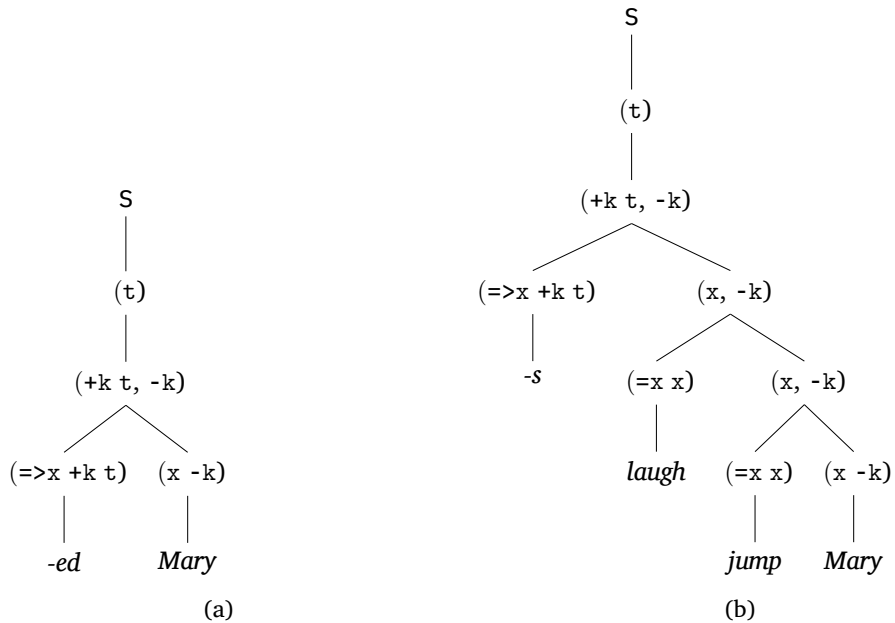


Figure 23: CFG parse trees: overgeneration by Figure 21c

DOUBLE OBJECT CONSTRUCTION
REVISITED

We will now take a step up from toy examples towards more interesting applications of the technique introduced above and re-examine the double object construction in the light of MDL. As pointed out in Section 1, there are two groups of approaches to sentences like *John gave Mary a book*: those which postulate a small clause complement of *give*, and those which maintain that the double object construction is monoclausal. Enumerating and analyzing all known arguments from both sides in a comprehensive way falls outside the scope of this paper. Instead, this section serves as proof of concept. In what follows, we convert a small sample of these arguments into the MG formalism and examine how the predictions of each analysis translate into higher or lower MDL values.

Let us focus on two facts regarding the English double object construction coming from two different sources. The first one is Harley and Jung (2015), who point out multiple parallels between double object structures with *give* and sentences with *have*. These are used to motivate an analysis where both *have* and *give* contain a possessive small clause headed by the abstract silent element P_{HAVE} . One of these parallels is an animacy restriction. Both possessors in *have*-clauses (1a, 1c) and Goal arguments in *give*-clauses (1b, 1d) are required to be animate, as long as the possession is alienable.

- (1) a. John has a book.
 b. Brenda gave John a book.
 c. #The car has a flyer.
 d. #The advertiser gave the car a flyer.

(Harley and Jung 2015, p. 704)

The second source is Kawakami (2018), who argues against the small-clause analysis, citing a number of discrepancies between the properties of known small clause constructions (e.g. *John considers Mary angry*) and those of *give*-clauses. One of the arguments supporting this stance comes from *wh*-movement and ambiguity. For sentences with *consider* (2a), both the matrix clause and the small clause can be

modified by *why*, yielding two different interpretations. On the other hand, the double object construction behaves as monoclausal, allowing only one reading where *why* modifies the matrix clause (2b).

- (2) a. Why did John consider Mary angry at Bill?
READING: asking the reason of considering
asking the reason of being angry
- b. Why did John give Mary a book?
READING: asking the reason of giving
#asking the reason of having
(Kawakami 2018, pp. 220–221)

Which of these two arguments is stronger with respect to encoding costs? We start by translating each of them into an MG. Assuming a consensus on all issues other than the double object construction, the two grammars should share most of their LIs. Since this example involves *wh*-movement, we consider complete expressions of category *c* rather than *t*. The shared lexical items are given in Figure 24a, and the additional LIs for *have* and *give* in the monoclausal and SC account are presented in Figure 24b and Figure 24c respectively.¹⁴ In accordance with the simplifying assumptions stated in Section 3, we ignore non-concatenative morphology and assume a separate set of morphological rules which realize *have-s* as *has* and *do-s* as *does*.

¹⁴These grammars rely on using multiple lexical items with ϵ as the string component. Such empty LIs have been widely used in MGs since their conception in Stabler 1997 and can be thought of as a method of compressing the grammar. Consider, for instance, $\epsilon :: =d_a +k d -k$, which allows any DP of category d_a to become a d , but not vice versa. The same restriction can be enforced without an empty LI by having two versions of each of its possible complements (*John* :: $d_a -k$, *Mary* :: $d_a -k$, *John* :: $d -k$, and *Mary* :: $d -k$), at the cost of introducing some redundancy into the lexicon.

More generally, empty LIs are how MGs express subcategorization requirements that are based on a hierarchy of projections rather than exact category matches. One alternative to this approach is an explicit hierarchy encoded as a partial order over selectors (Fowlie 2013) – although the cost of such an addition to the formalism would also need to be taking into account when calculating MDL. That said, certain empty LIs correspond to empty heads introduced in the theoretical literature and are therefore necessary to formalize them faithfully. For example, $\epsilon :: =d +k d_a = sc$ represents the empty element P_{HAVE} central to the analysis of Harley and Jung (2015).

<i>John</i> :: d _a -k	<i>consider</i> :: =sc V	<i>angry</i> :: a
<i>Mary</i> :: d _a -k	-ε :: =>V +k d= v	ε :: =a d= sc
<i>the car</i> :: d -k	-ε :: =>v x	<i>why</i> :: w -wh
<i>a flyer</i> :: d -k	<i>do</i> :: =x do	ε :: =sc w= sc
ε :: =d _a +k d -k	-ε :: =>x do	-ε :: =>t +wh c
	-s :: =>do +k t	ε :: =t c
(a) Shared lexical items		
		ε :: =d +k d _a = sc _{poss}
ε :: =d +k d _a = sc	ε :: =d +k d _a = sc	-ε :: =>sc _{poss} sc
<i>have</i> :: =sc v	<i>have</i> :: =sc v	<i>have</i> :: =sc _{poss} v
<i>give</i> :: =d +k d= V	<i>give</i> :: =sc V	<i>give</i> :: =sc _{poss} V
(b) Monoclausal <i>give</i>	(c) Uniform SC <i>give</i>	(d) Refined SC <i>give</i>

Figure 24: MG implementations of the double object construction

The simple solution in Figure 24c views all small clauses as having the same syntactic category, *sc*. This validates Kawakami’s (2018) objections to the small clause analysis based on multiple differences between small clauses selected by *consider* and arguments of *give*. However, Harley and Jung (2015, p. 718) point out a way to reconcile the two groups of phenomena, suggesting a typology of small clauses. Under this view, small clauses embedded under *consider* (unlike those under *give*) include an additional projection, which explains different properties. Translating this idea into MGs yields the set of LIs given in Figure 24d. Possessive small clauses (*sc_{poss}*) are selected by both *have* and *give*, and may merge with an empty LI to form expressions of category *sc*, which are selected by *consider*.

The animacy restriction is implemented by giving animate DPs a category feature distinct from *d*, *d_a*. An animate DP can freely become a normal DP by merging with ε :: =d_a +k d -k, but the opposite is not possible. In other words, *d_a* occurs in all contexts that allow *d*, and also in some contexts where *d* is prohibited. The restriction on modification by *why* is added by only allowing *why* to merge with small clauses – expressions of category *sc*. This is done via two LIs: *why* :: w -wh and ε :: =sc w= sc. This fragment allows *why* to modify

small clauses but not matrix clauses, since only the former are relevant for the example.¹⁵

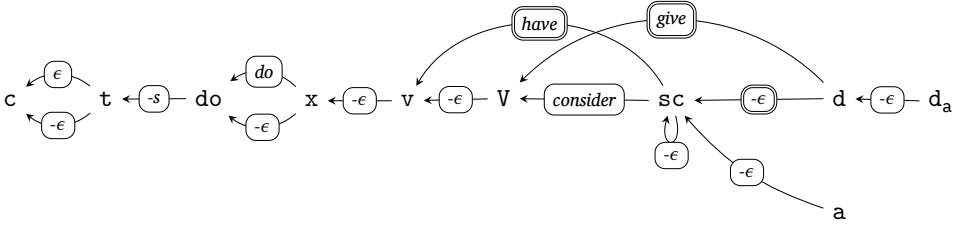
Note that all three grammars are associated with some overgeneration. First, there is no restriction requiring *do*-support in interrogative contexts, which gives rise to examples like **why consider-s John Mary angry*. In addition, all grammars except refined SC treat all small clauses as uniform, producing strings like **John have-s angry* (and, in the case of the uniform small clause analysis, **John give-s Mary angry*). As we have seen before, overgeneration does not affect grammar encoding, but will contribute to a higher cost of encoding some grammatical sentences.

Consider the head-complement graphs in Figure 25. The monoclausal *give* (Figure 25a) selects its arguments directly, whereas the uniform SC *give* (Figure 25b) takes as its complement the same small clause as *have* and shares its restriction on animacy. On the other hand, the loop at the *sc* vertex represents the position modifiable by *why*. The monoclausal *give* bypasses the category *sc*, unlike *have*; the latter, but not the former, is compatible with *why*. However, the uniform SC *have* merges with expressions of category *sc*, incorrectly allowing modification by *why*. Finally, the refined SC analysis (Figure 25c) gets around both problems by distinguishing between *sc* and *sc_{poss}*.

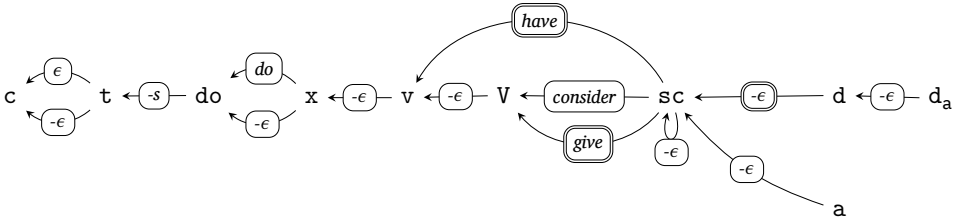
As a further illustration, some derived tree examples are given in Figure 26.

Grammar encoding costs (Table 4) reflect generalizations made by each grammar, as well as the number of category distinctions it makes. Both monoclausal and uniform SC approaches require 13 distinct categories; however, the latter has a lower cost as it reuses the abstract element heading a small clause, $\epsilon :: =d +k d_a = sc$, to provide arguments to both *have* and *give*. Refined SCs require an extra category, *sc_{poss}*, as well as an additional lexical item, $-\epsilon :: =sc_{poss} sc$, so this grammar ends up having the highest encoding cost.

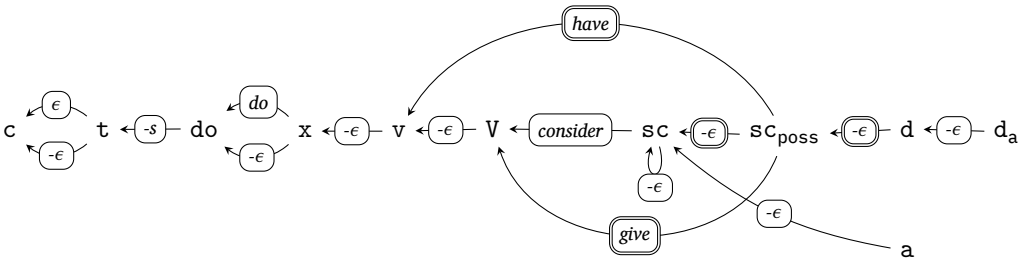
¹⁵For the sake of completeness, it would be easy to add modification of matrix clauses by introducing one more empty lexical item: $-\epsilon :: =>v w = v$. Then the grammar would generate different structures corresponding to different readings of *consider*-clauses: *why [do-s John consider [Mary angry] why]* vs. *why [do-s John consider [Mary angry why]]* (cf. item 2a).



(a) Monoclausal



(b) Uniform SC



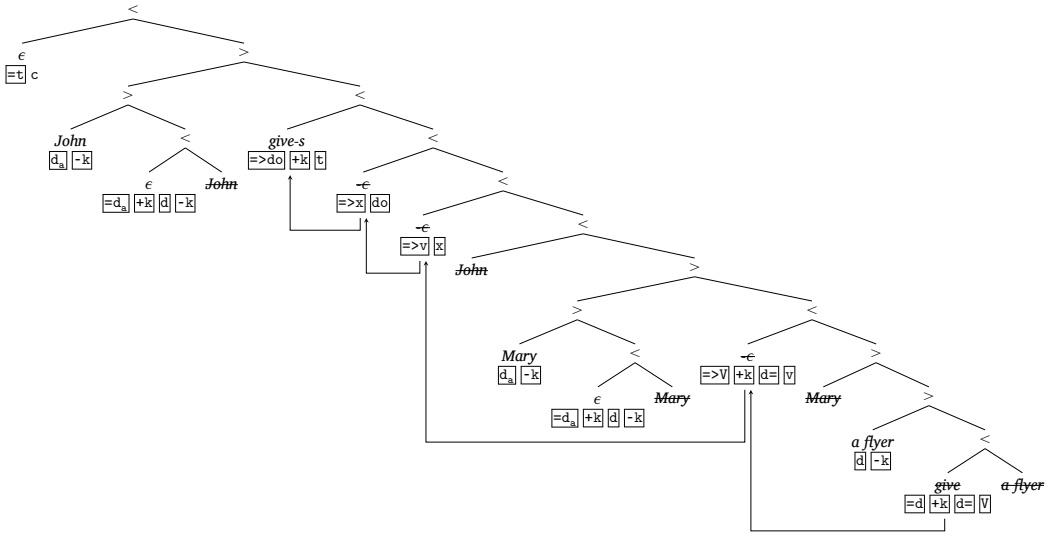
(c) Refined SC

Figure 25: Head-complement graphs of MGs in Figure 24; LIs not shared by all grammars are highlighted with double frames

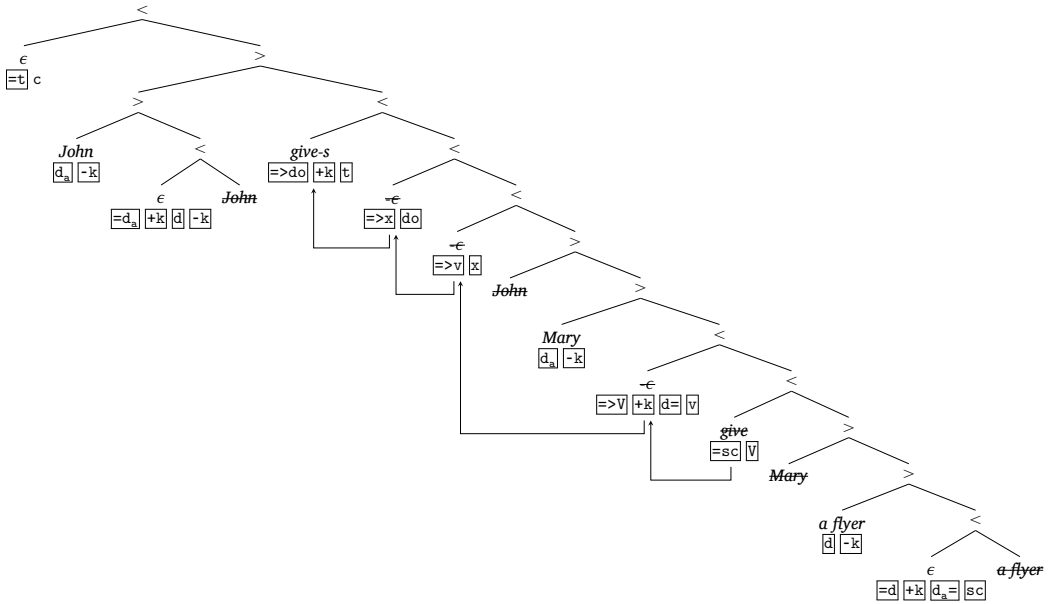
Table 4:
Grammar
metrics for the
double-object
construction

	$ Base $	\sum_{syn}	\sum_{phon}	Grammar (bits)
Monoclausal	13	51	50	955.39
Uniform SC	13	49	50	933.17
Refined SC	14	51	50	966.20

Evaluating syntactic proposals using MGs and MDL



(a) Monoclausal



(b) Uniform SC

Figure 26: Derived trees for *John give-s Mary a flyer*

Table 5:
Sentence
encoding costs
for the
double-object
construction
(bits)

	Monoclausal	Uniform SC	Refined SC
<i>John give-s Mary a flyer</i>	$6 \log_2 2 + 3 \log_2 3$ ≈ 10.75	$7 \log_2 2 + 2 \log_2 3$ ≈ 10.17	$6 \log_2 2 + 2 \log_2 3$ ≈ 9.17
<i>Mary have-s a flyer</i>	$5 \log_2 2 + \log_2 3$ ≈ 6.58	$5 \log_2 2 + \log_2 3$ ≈ 6.58	$4 \log_2 2 + \log_2 3$ ≈ 5.58
<i>John consider-s Mary angry</i>	$7 \log_2 2 + 2 \log_2 3$ ≈ 10.17	$7 \log_2 2 + 2 \log_2 3$ ≈ 10.17	$7 \log_2 2 + 2 \log_2 3$ ≈ 10.17
<i>why do-s John consider Mary angry</i>	$6 \log_2 2 + 2 \log_2 3$ ≈ 9.17	$7 \log_2 2 + 2 \log_2 3$ ≈ 10.17	$5 \log_2 2 + 2 \log_2 3$ ≈ 8.17

In order to see how individual analysis choices contribute to corpus encoding, consider the costs of four different sentences shown in Table 5. Note that these four sentences are not meant to represent the entire corpus (and we do not calculate the final corpus cost or MDL value for this case study), but rather to illustrate how various data points contribute to the differences between our grammars with respect to corpus cost. Partial CFGs are given in Figure 27; for space reasons, it only includes rules with non-zero cost, i.e. those which share the left-hand side with at least one other rule.

As expected, the monoclausal approach pays a higher cost to encode examples with *give*, because of its lack of animacy restrictions, whereas the uniform SC grammar overpays for grammatical sentences involving modification by *why*. The third option, refined SCs, does not overpay in either case. In addition, it pays a lower cost to encode *Mary has a flyer*, because of its distinction between small clause types. This corresponds to the fact that this grammar, unlike the other two, does not generate strings like **John has angry*.

For a closer look at individual rules' contribution to these values, let us examine detailed costs of encoding a double object construction, provided in Table 6. All three grammars must pay the cost of picking *a flyer* as the object. The monoclausal approach, which lacks animacy restrictions, pays the extra cost of picking an animate Goal, in the form of an additional use of $(d \ -k) \rightarrow (+k \ d \ -k, \ -k)$. Next, all three grammars use a rule to select the right complement type for the verb. However, since the uniform SC grammar assigns the same feature bundle to *give* and *consider*, it has to pay an additional bit to use $(=_{sc} \ V) \rightarrow give$ and pick the former. Refined SCs pay for each

$(d \text{ -}k) \rightarrow \textit{the car}$	
$(d \text{ -}k) \rightarrow \textit{a flyer}$	
$(d \text{ -}k) \rightarrow (+k \text{ d } -k, -k)$	
$(d_a \text{ -}k) \rightarrow \textit{John}$	
$(d_a \text{ -}k) \rightarrow \textit{Mary}$	
$(sc, -k) \rightarrow (d= sc) (d \text{ -}k)$	$(sc, -k) \rightarrow (d_a= sc) (d_a, -k)$
$(v, -k) \rightarrow (d= v) (d \text{ -}k)$	$(V, -k) \rightarrow (=sc V) (sc, -k)$
$(do, -k) \rightarrow (=x \text{ do}) (x, -k)$	$(V, -k) \rightarrow (d= V) (d \text{ -}k)$
$(do, -k) \rightarrow (=>x \text{ do}) (x, -k)$	$(v, -k) \rightarrow (=sc v) (sc, -k)$
$(c) \rightarrow (=t \text{ c}) (t)$	$(do, -k, -wh) \rightarrow (=x \text{ do}) (x, -k, -wh)$
$(c) \rightarrow (+wh \text{ c}, -wh)$	$(do, -k, -wh) \rightarrow (=>x \text{ do}) (x, -k, -wh)$
$(do, -wh, -k) \rightarrow (=x \text{ do}) (x, -wh, -k)$	$(t, -wh) \rightarrow (+k \text{ t}, -k, -wh)$
$(do, -wh, -k) \rightarrow (=>x \text{ do}) (x, -wh, -k)$	$(t, -wh) \rightarrow (+k \text{ t}, -wh, -k)$
(a) Shared rules	(b) Monoclausal
$(=sc V) \rightarrow \textit{give}$	
$(=sc V) \rightarrow \textit{consider}$	
$(sc, -k) \rightarrow (d_a= sc) (d_a, -k)$	
$(v, -k) \rightarrow (=sc v) (sc, -k)$	
$(do, -k, -wh) \rightarrow (=x \text{ do}) (x, -k, -wh)$	$(sc, -k) \rightarrow (=>sc_{\text{poss}} \text{ sc}) (sc_{\text{poss}}, -k)$
$(do, -k, -wh) \rightarrow (=>x \text{ do}) (x, -k, -wh)$	$(V, -k) \rightarrow (=sc V) (sc, -k)$
$(t, -wh) \rightarrow (+k \text{ t}, -k -wh)$	$(V, -k) \rightarrow (=sc_{\text{poss}} V) (sc_{\text{poss}}, -k)$
$(t, -wh) \rightarrow (+k \text{ t}, -wh -k)$	$(v, -k) \rightarrow (=sc_{\text{poss}} v) (sc_{\text{poss}}, -k)$
(c) Uniform SC	(d) Refined SC

Figure 27: Nonzero-cost CFG rules for Figure 24

Table 6:
Nonzero-cost
rules deriving
John give-s Mary
a flyer and their
costs (bits)

	Rule	Cost	Total
Shared rules	$(c) \rightarrow (=t\ c)\ (t)$	$\log_2 2$	≈ 6.58
	$(v, -k) \rightarrow (d=v)\ (d\ -k)$	$\log_2 2$	
	$(do, -k) \rightarrow (=x\ do)\ (x, -k)$	$\log_2 2$	
	$(d_a\ -k) \rightarrow \textit{John}$	$\log_2 2$	
	$(d_a\ -k) \rightarrow \textit{Mary}$	$\log_2 2$	
	$(d\ -k) \rightarrow \textit{a flyer}$	$\log_2 3$	
Monoclausal	$(d\ -k) \rightarrow (+k\ d\ -k, -k)$	$2\log_2 3$	≈ 4.17
	$(V, -k) \rightarrow (d=V)\ (d\ -k)$	$\log_2 2$	
Uniform SC	$(d\ -k) \rightarrow (+k\ d\ -k, -k)$	$\log_2 3$	≈ 3.58
	$(sc, -k) \rightarrow (d_a = sc)\ (d_a, -k)$	$\log_2 2$	
	$(=sc\ V) \rightarrow \textit{give}$	$\log_2 2$	
Refined SC	$(d\ -k) \rightarrow (+k\ d\ -k, -k)$	$\log_2 3$	≈ 2.58
	$(V, -k) \rightarrow (=sc_{\text{poss}}\ V)$	$\log_2 2$	

distinction only once, resulting in the lowest cost of encoding the sentence. Thus, the cost of this more complex grammar is offset by the lower cost of encoding the data.

Essentially, what the two positions exemplified by Harley and Jung 2015 and Kawakami 2018 disagree on is exactly what properties *have* shares with *give*. MGs can represent these shared properties as syntactic features within LIs which are reused in multiple constructions. The technique outlined here offers a way to examine and directly compare insights from multiple literature sources while accounting for possible overgeneration.

6

DISCUSSION AND FUTURE WORK

We have investigated the possibility of comparing syntactic analyses on quantitative grounds. Even within the same framework, such as Chomsky’s (1995, 2000) Minimalist Program, there is enough room for alternative accounts of the same observed language data. We have shown how specific proposals stated as minimalist grammars (Stabler

1997) can be compared with the help of an evaluation measure inspired by minimum description length (Rissanen 1978), and how different predictions made by these proposals translate into quantifiable differences.

Examples throughout the paper have demonstrated how, overall, correct generalizations lead to smaller grammars, while overgeneration increases the corpus cost. The case study of the double-object construction presented here is a proof of concept demonstrating how MDL can offer a quantitative perspective on various issues that are a matter of debate, or simply a topic of interest, for syntacticians. A few potential examples are listed below.

Hierarchy of adjectives vs. unordered adjuncts: One could ask whether complex cartography-style structures are “worth it” for a given set of data. For instance, Bayırlı (2018) argues that Turkish adjectives obey the adjective hierarchy of Cinque (1994) and Cinque (2010), whereas Grashchenkov and Isaeva (2023) suggest a lack of strict ordering of adjectives having used a corpus of Turkish and other Turkic data in their study. From the MDL perspective, choosing to implement a hierarchy of adjectives (through empty LIs or as a Fowlie (2013)-style extension to standard MGs) would carry the cost of encoding it. Conversely, allowing adjectives in any order may lead the grammar to overgenerate, increasing the cost of encoding adjective orderings that *are* attested in the corpus.

Acategorical roots: The idea of roots being category-neutral and having to merge with a categorizing head is a general assumption in Distributed Morphology (Marantz 1997; Embick and Marantz 2008). An MG implementation of this strategy would have root LIs carrying a single category feature and categorizing heads as LIs selecting expressions of that category, as opposed to having multiple lexical items with the same root. Quantitatively, we can expect this to be beneficial for the grammar cost, as long as the number of roots is large enough to justify the cost of the extra features needed to merge the roots and categorizing heads.

***There-insertion* in English:** Ermolaeva and Kobele (2022) use an MG-like formalism to compare various analyses of expletive-*there* constructions in English, describing their differences in terms of syntactic dependencies; MDL could translate these differences into quantitative terms. The high-origin account, where *there* merges directly into the

specifier of TP (Chomsky 2000), requires fewer LIs and syntactic features to encode than one where *there* merges low and moves to its surface position (Deal 2009; Alexiadou and Schäfer 2011) or starts out in a constituent with its associate (Basilico 1997; Sabel 2000). At the same time, it would have trouble expressing any restrictions on lexical verbs (*There arrived a man in the station* vs. *There laughed a man in the hallway*; see Deal 2009), leading to overgeneration and consequences to the corpus cost.

All of that said, there is plenty of room for refinement. MDL can disagree with a linguistic intuition on what constitutes a simpler explanation of the data, if some aspect of the analysis is not taken into account by the encoding scheme, or cannot be expressed by the chosen formalism, or requires an overhead cost that does not pay off in the case of the chosen corpus. Let us briefly discuss each of these variables in turn.

Encoding schemes: The method of encoding MGs defined in Section 4, where each LI is considered a sequence of symbols from the same encoding table, is straightforward but naive. A few of the possible modifications include switching from the fixed-length code presented here to a variable-length code, with shorter binary sequences for more frequent symbols (Lee 2001); rethinking how elements of LIs are parsed into symbols to be encoded (for instance, it may be useful to combine the type and category of syntactic features into a single symbol); encoding each string component and/or feature bundle only once and using pointers to refer to them (Xanthos *et al.* 2006) to incentivize (i.e. lower the cost of) reusing elements that have already been introduced. Some proposals in the linguistics literature are motivated by patterns that could only be translated into cost reduction under a sophisticated encoding scheme; see Appendix A for an example.

Formalism-related choices: The version of minimalist grammars defined in Section 3 is a relatively simple but detailed one for the sake of conceptual clarity. Treating a lexical item as a string of phonological segments (approximated by orthography) followed by syntactic features is explicitly a simplification, as is using head movement as the sole operation for building complex words; but a more sophisticated formalism could be used to bring the results closer to those of theoretical syntax.

In order to compare analyses constructed with different formal machinery in mind (or to compare different formalisms), one would have to consider the cost of encoding this machinery along with the grammar and corpus. For example, compare MGs as defined in this paper vs. a version without covert movement. The grammar cost of an analysis using the latter formalism might be lower because of fewer distinct symbols involved; or it might be higher due to additional lexical items needed to compensate for the missing machinery. At the same time, the two versions would also differ in how **merge** and **move** are defined, the latter being cheaper to encode as it lacks the syntactic operation of covert movement. Chater *et al.* (2015) goes even further, proposing a higher-order version of MDL computed as the sum of four terms representing encoding lengths of (i) a Turing machine capable of describing Universal Grammars; (ii) a Universal Grammar (\approx formalism) capable of stating specific grammars; (iii) a grammar generating the given corpus; and (iv) the corpus as encoded by the grammar.

Corpora: As shown in Footnote 5, extremely small datasets can favor overfitting grammars, if the reduction in corpus cost provided by introducing syntactic generalizations is insufficient to justify the initial investment in the grammar. This also applies to large but repetitive datasets; an extreme case would be a corpus containing the same sentence repeated an arbitrarily large number of times. Conversely, with a very large corpus of diverse sentences (which is a better representation of natural language as a whole) the MDL value is decided primarily by the corpus cost. At the same time, the grammar cost still contributes to the choice of the grammar, since many distinctions between grammars (that a linguist would consider important) have no effect on corpus cost. Consider a set of m roots, each compatible with any of n suffixes, for the total of $m \times n$ words, all found in the same syntactic contexts and attested in the corpus.¹⁶ A minimalist grammar can encode these as whole words, with $m \times n$ lexical items, or as separate roots and suffixes, resulting in $m + n$ lexical items carrying shorter string components. On the corpus side, the former option

¹⁶This generalizes the observation illustrated by the toy grammar in Section 4, where $m = 2$ and $n = 2$; it is easy to see how this would scale with more lexical verbs in the corpus.

corresponds to a single choice out of $m \times n$ options, costing $\log_2(m \times n)$ bits. The latter requires picking the root and the suffix separately, for $\log_2(m) + \log_2(n) = \log_2(m \times n)$ bits. The corpus cost is the same for both options, whereas the grammar cost may be significantly different.

More fundamentally, linguists put a lot of emphasis on obtaining independent evidence for their proposals to justify the theoretical cost of postulating a new structure or operation. In an informal setting, this evidence would be brought in as a set of examples. With the proposal translated into a formal grammar, reusing a lexical item in multiple structures translates into a measurable reduction to the grammar cost, while failure to capture an observed contrast would lead to overgeneration and result in a higher corpus cost. If a consensus is reached on the set of data we care about (the corpus), and if we fix or take into account what shape analyses may take (the formalism) and how they are quantified (the encoding scheme), we can keep track of the strength of every relevant argument and counter-argument.¹⁷ The minimum description length principle works as a natural evaluation measure, bringing the notion of “intuitive goodness” of syntactic descriptions a step closer to the more easily definable notion of “quantitative goodness”.

ACKNOWLEDGMENTS

The author was supported by the Fellowship from Non-commercial Foundation for the Advancement of Science and Education INTELLECT.

¹⁷A reviewer has noted that there is little agreement in mainstream Minimalism with respect to the corpus and formalism, and a consensus has been impossible to reach so far. While this is a very valid concern, the MDL-based approach does not create a new problem but rather highlights one already present. The literature is replete with different (sometimes incompatible) assumptions of what grammars are allowed to look like, beliefs regarding the content of the universal grammar, and analyses developed for overlapping but non-identical datasets. The approach outlined in this paper makes this problem explicit, defining more precisely what needs to be settled in order to solve it.

APPENDIX

A

The case study in Section 5 examines the distinction between the ditransitive verb selecting its arguments directly vs. selecting a constituent that is also found in other constructions and shares some properties with them. In what follows, we sketch a comparison along another dimension – namely, whether the double-object construction (*give Mary a book*) is related to the *to*-dative (*give a book to Mary*), illustrated by the original VP-shell analysis of Larson (1988) and the refined small clauses of Harley and Jung (2015).

Larson (1988) postulates a relation between the two constructions in question. Under his analysis, the structure of the VP containing the internal arguments of a ditransitive verb is parallel to that of a clause, with the Theme (*a letter*) corresponding to the subject and the Goal (*Mary*) to the object. The double object construction is derived from the *to*-dative via an operation analogous to passivization.

In order to see how this can be formalized, let us first implement passives in MGs. We start with the lexicon from Figure 24a (repeated in Figure 29a) and add two new LIs: *-ed* :: =>V pass and *be* :: =pass v (adapted from Kobele 2006). Then the passive construction is derived as shown in Figure 28. The expression of category V, with its topmost DP *Mary* still carrying its *-k* feature, is merged with *-ed* :: =>V pass, and the result with *be* :: =pass v. A subject is never merged in; instead, *Mary* is promoted to the subject position by having its *-k* checked by *-s* :: =>do +k t.

There are two issues preventing a faithful translation of Larson's (1988) solution into a minimalist grammar. First, the SMC requires that the movement of one argument be resolved before merging in another carrying the same licensee feature. A structure such as *[[give Mary] a letter]* with both DPs still carrying an unchecked *-k* would violate the SMC. We can bypass this problem (in a somewhat unsatisfying way) by constructing a version of *a letter* with its *-k* feature already checked. Second, the original analysis treats *to* as an instance of Case marking, which cannot be expressed in terms of standard MGs.¹⁸ With the exception of *to*-as-Case, this analysis

¹⁸This aspect of the analysis is out of reach for basic MGs but could be captured by an extended version of the formalism. See e.g. Ermolaeva 2018 and Ermolaeva

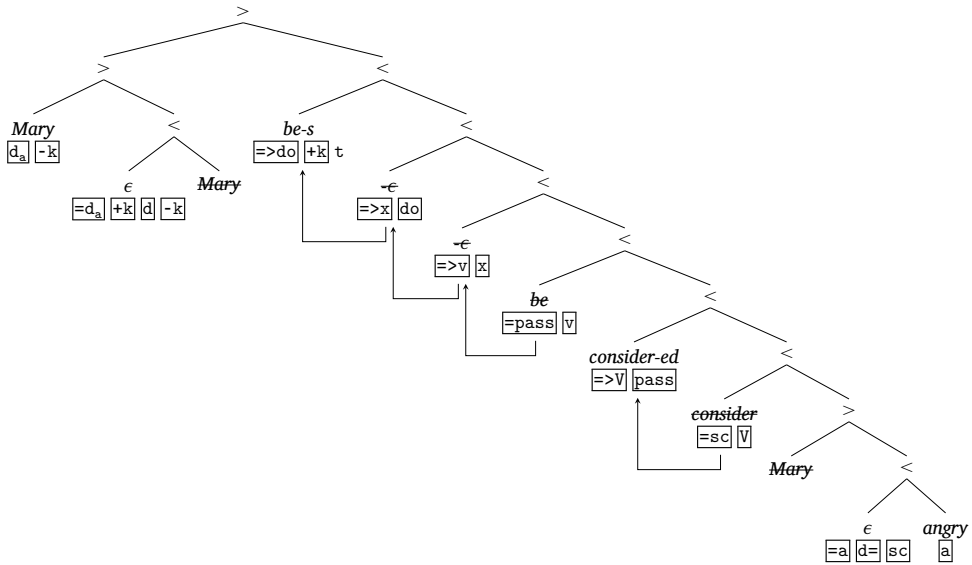


Figure 28: Derived tree for *Mary be-s consider-ed angry*

<i>John</i> :: d _a -k	<i>consider</i> :: =sc V	<i>angry</i> :: a
<i>Mary</i> :: d _a -k	-ε :: =>V +k d= v	ε :: =a d= sc
<i>the car</i> :: d -k	-ε :: =>v x	<i>why</i> :: w -wh
<i>a flyer</i> :: d -k	<i>do</i> :: =x do	ε :: =sc w= sc
ε :: =d _a +k d -k	-ε :: =>x do	-ε :: =>t +wh c
	-s :: =>do +k t	ε :: =t c

(a) Shared lexical items (= 24a)

	ε :: =d +k d _a = sc _{poss}	
	ε :: =d +k d _a = sc	-ε :: =>sc _{poss} sc
<i>have</i> :: =sc v	<i>have</i> :: =sc _{poss} v	
<i>give</i> :: =d y	<i>give</i> :: =sc _{poss} V	
ε :: =d +k d _t	<i>to</i> :: p	
-ed :: =>V pass	ε :: =d +k p= pp	
<i>be</i> :: =pass v	<i>give</i> :: =pp d= V	
-ε :: =>y +k d= V		
-ε :: =>y =d _t V		

(b) Shared passives

(c) Quasi-Larsonian *give*

(d) Extended refined SC *give*

Figure 29: MG implementations of the double object construction and *to*-datives

is recreated in Figure 29c; we will refer to this grammar as “quasi-Larsonian” to acknowledge its limitations. There is one lexical item, *give* :: =d y, shared by both constructions, which takes the Goal argument as its complement. To form a *to*-dative, the result then merges with $-\epsilon$:: =>y +k d= V, checking the Goal’s -k and merging in the Theme argument. To form a double object construction, it merges instead with $-\epsilon$:: =>y =d_t V, which selects the Theme argument with its -k already checked, leaving the Goal’s -k to be checked later in the derivation – similar to the object in a passive construction.

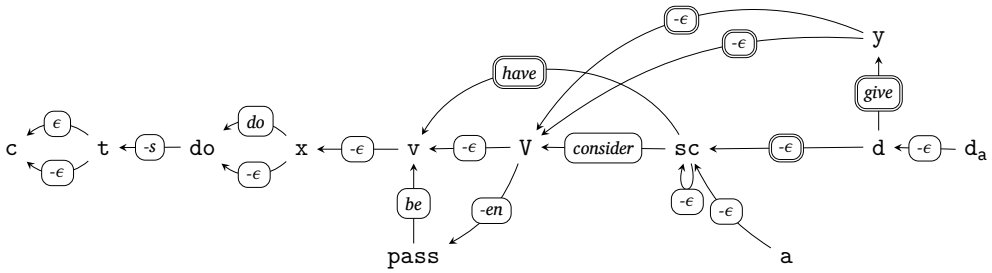
In contrast, Harley and Jung (2015) cite the approach to *to*-datives proposed in Harley 2007, which treats them as separate from the double-object construction, with the verb base-generated low in the structure. This solution can be translated into MGs by adding a separate version of *give* (which selects a prepositional phrase and a DP), as well as a method of constructing PPs. We refer to the result, given in Figure 29d, as “extended refined SCs”.

Both sets of LIs are compatible with the passive (Figure 29b) and ensure the promotion of the correct argument to the subject position (*Mary was given a letter* vs. *A letter was given to Mary*). Head-complement graphs for both grammars are given in Figure 30.

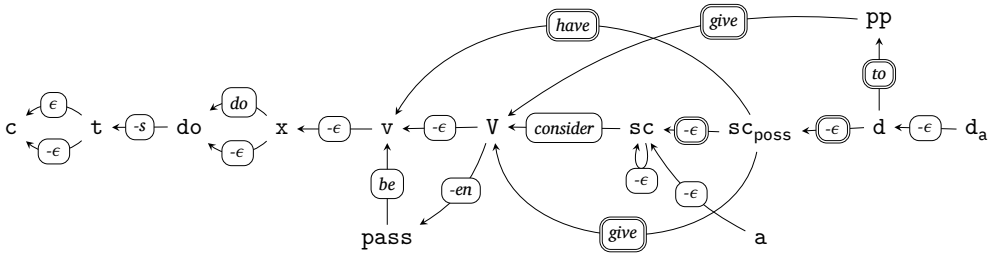
Let us first revisit the data points from Section 5 and assess their impact on the corpus cost. The quasi-Larsonian analysis performs largely like the monoclausal one. With respect to *why*-modification, no small clause in the constructions in question means no over-generation.¹⁹ The animacy restriction on the Goal is an argument against the quasi-Larsonian solution, since it applies to the double

and Kobele 2022 for an MG-compatible treatment of Agree as transmission of morphological information along syntactic dependencies. In their framework, *to* could be implemented as data transmitted to the Goal when the $-\epsilon$:: =>y +k d= V checks its -k feature.

¹⁹For completeness, the addition of passives does by itself affect some sentences we have *not* considered. The CFG for the quasi-Larsonian solution involves the same feature configuration on the left-hand side of two rules, $(v, -k, -wh) \rightarrow (=pass\ v)\ (pass, -k, -wh)$ and $(v, -k, -wh) \rightarrow (=sc\ v)\ (sc, -k, -wh)$, as it generates both *why be-s Mary considered angry* and *why do-s Mary have a flyer* (with *why* modifying the small clause). This applies to the monoclausal solution as well. The (extended) refined SC approach simply does not generate the latter of these sentences, so the remaining rule has a zero cost.



(a) Quasi-Larsonian



(b) Extended refined SC

Figure 30: Head-complement graphs of MGs in Figure 29

object construction and the *to*-dative to a different extent (Oehrle 1976). This is not a problem for extended refined SCs. However, the quasi-Larsonian solution relies on the LI which introduces the Goal, *give* :: =d y, being the same in both constructions – which is incompatible with the Goal animacy contrast between the two. While the MG fragments presented here simplify the contrast to “no restriction” vs. “animate only”, the same logic would apply to more nuanced distinctions. In terms of corpus cost, this instance of overgeneration means that the quasi-Larsonian analysis would overpay for each double object construction in the corpus, same as the monoclausal analysis.

For the grammar cost, a proper comparison is impossible without tweaking the formalism itself, due to the limitations discussed above. As is, the quasi-Larsonian grammar is shorter than the extended refined SC one (with the caveat that the former would also need to pay the cost of encoding *to*). This difference is small enough to be negated if we argue that the implementation of PPs should be shared by both grammars (as the LIs *to* :: p and ϵ :: =d +k p= pp would be required

for other constructions involving PPs). On the other hand, our basic encoding scheme is unable to take into account some elements of Larson's analysis – in particular, the parallel between passivization and the operation deriving the double object construction from the *to*-dative. This is reflected in the MG, for instance, by the similarities between $-\epsilon :: =>V +k d= v$ (which checks the object's $-k$ and merges in the subject to derive the active construction) and $-\epsilon :: =>y +k d= V$ (which checks the Goal's $-k$ and merges in the Theme to derive the *to*-dative). Both LIs are of the form $-\epsilon :: =>_ +k d= _$, with $_$ standing for the two category names that constitute their differences. A more refined encoding scheme capable of reusing, rather than reencoding, repeated *parts* of lexical items would be able to capitalize on this.

REFERENCES

- Artemis ALEXIADOU and Florian SCHÄFER (2011), There-insertion: An unaccusativity mismatch at the syntax-semantics interface, online proceedings of West Coast Conference on Formal Linguistics 28.
- Karlos ARREGI and Asia PIETRASZKO (2018), Generalized head movement, in Patrick FARRELL, editor, *Proceedings of the Linguistic Society of America*, volume 3, pp. 1–15.
- David BASILICO (1997), The topic is “there”, *Studia Linguistica*, 51(3):278–316.
- İsa Kerem BAYIRLI (2018), Does Turkish have adjective ordering restrictions?, *IULC Working Papers*, 18(2):1–26.
- Michael BRODY (2000), Mirror theory: Syntactic representation in perfect syntax, *Linguistic Inquiry*, 31(1):29–56.
- Nick CHATER, Alexander CLARK, John GOLDSMITH, and Amy PERFORS (2015), Towards a new empiricism for linguistics, in *Empiricism and Language Learnability*, pp. 58–105, Oxford University Press, Oxford, UK.
- Noam CHOMSKY (1956), Three models for the description of language, *IRE Transactions on Information Theory*, 2(3):113–124.
- Noam CHOMSKY (1957), *Syntactic structures*, De Gruyter Mouton, The Hague, Netherlands.
- Noam CHOMSKY (1965), *Aspects of the theory of syntax*, MIT Press, Cambridge, MA.

- Noam CHOMSKY (1986), *Knowledge of language: Its nature, origin, and use*, Praeger, New York, NY.
- Noam CHOMSKY (1995), *The minimalist program*, MIT Press, Cambridge, MA.
- Noam CHOMSKY (2000), Minimalist Inquiries: the framework, in Roger MARTIN, David MICHAELS, and Juan URIAGEREKA, editors, *Step by Step: Essays on Minimalist Syntax in Honor of Howard Lasnik*, pp. 89–156, MIT Press, Cambridge, MA.
- Noam CHOMSKY and Morris HALLE (1968), *The sound pattern of English*, Harper & Row, New York, NY.
- Guglielmo CINQUE (1994), On the evidence for partial N-movement in the Romance DP, in Guglielmo CINQUE, Jan KOSTER, Jean-Yves POLLOCK, and Rafaella ZANUTTINI, editors, *Paths towards universal grammar: Studies in Honor of Richard S. Kayne*, pp. 85–110, Georgetown University Press, Washington, DC.
- Guglielmo CINQUE (2010), *The syntax of adjectives: A comparative study*, MIT Press, Cambridge, MA.
- Alexander CLARK (2013), Learning trees from strings: A strong learning algorithm for some context-free grammars, *Journal of Machine Learning Research*, 14:3537–3559.
- Alexander CLARK (2015), Canonical context-free grammars and strong learning: two approaches, in Marco KUHLMANN, Makoto KANAZAWA, and Gregory M. KOBELE, editors, *Proceedings of the 14th Meeting on the Mathematics of Language (MOL 2015)*, pp. 99–111, Association for Computational Linguistics, Chicago, IL.
- Amy Rose DEAL (2009), The origin and content of expletives: Evidence from “selection”, *Syntax*, 12(4):285–323.
- Marina ERMOLAEVA (2018), Morphological agreement in minimalist grammars, in *Formal Grammar: 22nd International Conference, FG 2017, Toulouse, France, July 22-23, 2017, Revised Selected Papers*, pp. 20–36, Springer, Berlin, Germany.
- Marina ERMOLAEVA (2021), *Learning syntax via decomposition*, Ph.D. thesis, University of Chicago, Chicago, IL.
- Marina ERMOLAEVA and Gregory M. KOBELE (2022), Agree as information transmission over dependencies, *Syntax*, 25(4):466–507.
- Meaghan FOWLIE (2013), Order and optionality: Minimalist grammars with adjunction, in *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pp. 12–20, Association for Computational Linguistics, Sofia, Bulgaria.
- John GOLDSMITH (1980), Meaning and mechanism in grammar, *Harvard studies in syntax and semantics*, 3:423–449.
- John GOLDSMITH (2001), Unsupervised learning of the morphology of a natural language, *Computational Linguistics*, 27(2):153–198.

- John GOLDSMITH (2006), An algorithm for the unsupervised learning of morphology, *Natural Language Engineering*, 12(4):353–372.
- John GOLDSMITH (2011), The evaluation metric in generative grammar, presented at 50th Anniversary of the MIT Linguistics Department, Cambridge MA, December 2011.
- Thomas GRAF (2013), *Local and transderivational constraints in syntax and semantics*, Ph.D. thesis, UCLA, Los Angeles, CA.
- Pavel GRASHCHENKOV and Ulyana ISAEVA (2023), Vzaimnoe raspolozhenie atributivnyh prilagatel'nyh po dannym tyurkskih tekstov [Adjectival ordering on the data of Turkic texts], *Uralo-altayskie issledovaniya*, 48(1):22–32.
- Peter D. GRÜNWARD (2007), *The minimum description length principle*, MIT Press, Cambridge, MA.
- John T. HALE and Edward P. STABLER (2005), Strict deterministic aspects of minimalist grammars, in Philippe BLACHE, Edward STABLER, Joan BUSQUETS, and Richard MOOT, editors, *Logical Aspects of Computational Linguistics. LACL 2005. Lecture Notes in Computer Science*, pp. 162–176, Springer, Berlin, Germany.
- Heidi HARLEY (2002), Possession and the double object construction, *Linguistic Variation Yearbook*, 2(1):31–70.
- Heidi HARLEY (2007), The bipartite structure of verbs cross-linguistically (or: Why Mary can't exhibit John her paintings), in Thaïs Cristóforo SILVA and Heliana MELLO, editors, *Conferências do V Congresso Internacional da Associação Brasileira de Linguística*, pp. 45–84, Belo Horizonte, Brazil.
- Heidi HARLEY and Hyun Kyoung JUNG (2015), In support of the P_{HAVE} analysis of the double object construction, *Linguistic Inquiry*, 46(4):703–730.
- Yu HU, Irina MATVEEVA, John GOLDSMITH, and Colin SPRAGUE (2005), Using morphology and syntax together in unsupervised learning, in *Proceedings of the Workshop on Psychocomputational Models of Human Language Acquisition*, pp. 20–27, Association for Computational Linguistics, Ann Arbor, MI.
- Mark JOHNSON (2017), Marr's levels and the minimalist program, *Psychonomic Bulletin & Review*, 24(1):171–174.
- Roni KATZIR (2014), A cognitively plausible model for grammar induction, *Journal of Language Modelling*, 2(2):213–248, doi:10.15398/jlm.v2i2.85, <https://jlm.ipipan.waw.pl/index.php/JLM/article/view/85>.
- Masahiro KAWAKAMI (2018), Double object constructions: Against the small clause analysis, *Journal of Humanities and Social Sciences*, 45:209–226.
- Richard S. KAYNE (1984), *Connectedness and binary branching*, Foris Publications, Dordrecht, Netherlands.
- Richard S. KAYNE (1994), *The antisymmetry of syntax*, MIT Press, Cambridge, MA.

- Gregory M. KOBELE (2002), Formalizing mirror theory, *Grammars*, 5(3):177–221.
- Gregory M. KOBELE (2006), *Generating copies: An investigation into structural identity in language and grammar*, Ph.D. thesis, UCLA, Los Angeles, CA.
- Gregory M. KOBELE (to appear), Minimalist grammars and decomposition, in Kleantes K. GROHMANN and Evelina LEIVADA, editors, *The Cambridge Handbook of Minimalism*, Cambridge University Press, Cambridge, MA.
- Richard K. LARSON (1988), On the double object construction, *Linguistic Inquiry*, 19(3):335–391.
- Thomas C.M. LEE (2001), An introduction to coding theory and the two-part minimum description length principle, *International Statistical Review*, 69(2):169–183.
- David MARR (1982), *Vision: A computational investigation into the human representation and processing of visual information*, Henry Holt and Co., New York, NY.
- Jens MICHAELIS (1998), Derivational minimalism is mildly context-sensitive, in Michael MOORTGAT, editor, *International Conference on Logical Aspects of Computational Linguistics*, pp. 179–198, Springer, Berlin, Germany.
- Richard Thomas OEHRLE (1976), *The grammatical status of the English dative alternation*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.
- Christopher PEACOCKE (1986), Explanation in computational psychology: Language, perception and level 1.5, *Mind & Language*, 1(2):101–123.
- David Michael PESETSKY (1996), *Zero syntax: Experiencers and cascades*, MIT Press, Cambridge, MA.
- Ezer RASIN, Iddo BERGER, Nur LAN, and Roni KATZIR (2018), Learning phonological optionality and opacity from distributional evidence, in Sherry HUCKLEBRIDGE and Max NELSON, editors, *Proceedings of the North East Linguistic Society 48 (NELS 48)*, volume 48, pp. 269–282, Amherst, MA.
- Ezer RASIN and Roni KATZIR (2016), On evaluation metrics in optimality theory, *Linguistic Inquiry*, 47(2):235–282.
- Ezer RASIN and Roni KATZIR (2019), Simplicity-based learning in constraint-based and rule-based phonology, mini-course at the University of Leipzig.
- Jorma RISSANEN (1978), Modeling by shortest data description, *Automatica*, 14(5):465–471.
- Joachim SABEL (2000), Expletives as features, in Roger BILLEREY and Brook Danielle LILLEHAUGEN, editors, *Proceedings of the 19th West Coast Conference on Formal Linguistics*, pp. 411–424, Cascadilla Press, Somerville, MA.

- Stuart M. SHIEBER (1985), Evidence against the context-freeness of natural language, *Linguistics and Philosophy*, 8:333–343.
- Edward P. STABLER (1997), Derivational minimalism, in Christian RETORÉ, editor, *Logical Aspects of Computational Linguistics: First International Conference, LACL '96 Nancy, France, September 23–25, 1996 Selected Papers*, pp. 68–95, Springer, Berlin, Germany.
- Edward P. STABLER (2001), Recognizing head movement, in *Proceedings of the 4th International Conference on Logical Aspects of Computational Linguistics, LACL '01*, pp. 245–260, Springer-Verlag, Berlin, Germany.
- Edward P. STABLER and Edward L. KEENAN (2003), Structural similarity within and among languages, *Theoretical Computer Science*, 293(2):345–363.
- John TORR and Edward STABLER (2016), Coordination in minimalist grammars: Excorporation and across the board (head) movement, in David CHIANG and Alexander KOLLER, editors, *Proceedings of the 12th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ 12)*, pp. 1–17, Düsseldorf, Germany.
- Edwin S. WILLIAMS (1975), Small clauses in English, in John P. KIMBALL, editor, *Syntax and Semantics, volume 4*, pp. 249–273, Brill, Leiden, Netherlands.
- Aris XANTHOS, Yu HU, and John GOLDSMITH (2006), Exploring variant definitions of pointer length in MDL, in *Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology and Morphology at HLT-NAACL 2006*, pp. 32–40, Association for Computational Linguistics, New York, NY.

Marina Ermolaeva

© 0000-0001-7796-7963


mail@ermolaeva.com

Lomonosov Moscow State University
Department of Theoretical and
Applied Linguistics
1-51 Leninskie Gory, Moscow, Russia

Marina Ermolaeva (2023), *Evaluating syntactic proposals using minimalist grammars and minimum description length*, *Journal of Language Modelling*, 11(1):67–119

doi <https://dx.doi.org/10.15398/jlm.v11i1.334>

This work is licensed under the *Creative Commons Attribution 4.0 Public License*.

cc  <http://creativecommons.org/licenses/by/4.0/>