Minimalist Grammars and Decomposition

Greg Kobele

April 12, 2021

In an exchange with Pullum, Chomsky [1990] writes

> Theories should be formulated clearly enough, and observations firmly enough established, so that inquiry can proceed in a constructive way. Beyond that, experiments can be carried out more carefully and theories made more precise, but the burden of proof is on those who consider the exercise worth undertaking.

As Chomsky points out, it is the person who deviates from standard practice in a community that must convince others of the value of their position. Many arguments have been marshalled in support of adopting a more rigorous stance than is common in Minimalist syntax. A precise linguistic formalism allows us to make rigorous contact with other fields, most notably in the behavioural and neural sciences, to reveal fundamental similarities and differences between different *linguistic* theories, and to formulate typological universals and proposing on that basis novel empirical research questions, not to speak of its indespensibility for more engineering based applications. Some of these will be taken up in Graf's chapter in this volume. However, these are justifications which do not address the usual concerns of the working linguist, such as 'how do I analyze this data' or 'how do I adjudicate between these two theoretical proposals'? Knowing that there exists an algorithmic way of converting an analysis that uses only leftward movement into to one that uses only rightward movement is a small consolation to a linguist who is asking not whether an analysis is *possible*, but whether it is *good*. Still, a precise formalism is useful as well to working linguists, precisely because it *does* allow for their questions to be addressed, and this in sometimes novel ways. The aim of this chapter is to introduce the formalism of Minimalist grammars, and to demonstrate that it can offer useful insights into linguistics.

Chomsky has said repeatedly that Minimalism is a *program*, and not a theory, thus emphasizing that there is a coherent research program, indepen-

dent of any particular instantiation of it.[1] Thus it makes perfect sense to have Starke-style Nanosyntax coexist alongside the more spare Chomskyian C-T-v-V clausal backbone; both can be different (perhaps even incompatible) instantiations of the same general research program. Similarly, a formalisation of Minimalism will of necessity cleave to a particular instance of the program. Yet a good formalisation will also be able to be modified so as to formalise different instances within the program. Ed Stabler formalized an early version of Minimalism in the late nineties [Stabler, 1997]. From the standpoint of the present, it looks somewhat baroque: there is no discussion of agreement, Merge is not free, there are no copies, no set-theoretic structures, no intepretable/uninterpretable distinction, no indices on traces, no sidewards movement, no counter-cyclic operations, no tucking-in, no structure removal, etc. Yet it has been repeatedly shown that the gamut of analytical variations can be straightforwardly added to this basic formalism (see e.g. Graf [2012] for an overview). Stabler [2011] has demonstrated the extensibility of the Minimalist grammar formalism, extending it with (un-)interpretable features, relativized minimality and phase-like constraints on movement, as well as providing pointers to the literature for other extensions (for example on various forms of head-movement, adjunction, sidewards movement, and copying).

There are innumerable (sometimes slight) variations of theories in the Minimalist vein. Many variants seem to diverge from each other at particular points:

1. What are the syntactic atoms?
2. How does syntax relate to the interfaces?
3. Where does explanation reside?

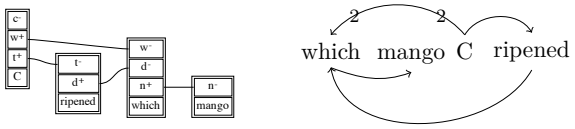
The answers to these questions pursued in this chapter are:

1. The syntactic atoms are *feature towers*.
2. Derivations are *directly interpreted*.
3. The locus of linguistic analysis is *the lexicon*.

[1] Chomsky characterizes the research program underlying Minimalism in terms of an approach to what he calls 'Darwin's Problem' (the evolutionary emergence of language). This program is then wholly independent of the grammatical architecture, of movement, of agreement, probes and goals, and what not. This is not what I am characterizing here as 'the program of Minimalism,' which starts from merge and move as the basic grammatical operations (as is Chomsky's actual practice).

One prominent difference between Minimalist grammars and current Minimalist practice lies in the fact that Minimalist grammars emphasize the lexicon, while Minimalist practice tends to emphasize the interfaces. Graf [2011] and Kobele [2011] have proven a result which entails that it is formally possible to shift the filtering effects of the interfaces into the lexicon, and to shift the derivational control of the lexicon over to the interfaces (this is discussed in Kobele [2014], and greatly expanded upon in Graf [2017]). There is surely some compromise of optimal elegance, rendering unto the interface what is the interface's, and unto the lexicon what is the lexicon's. I will emphasize the lexicon in this contribution, thereby compensating for current Minimalist practice.

Despite this emphasis on the lexicon, many points will emerge which are completely independent thereof. For example, while we will begin our analysis with *matching graphs over feature towers*, as in figure 1(a), indicating which features of which lexical items check each other during a derivation, we will construct our analyses using *dependency structures*, which make no reference to features at all. (Fear not, these representations are equivalent, both to each other and to more familiar constituentful structures, as will be shown in section 1.3.) When representations are equivalent, they are not al-



(a) Matching graphs over feature towers

(b) Dependency structures

$$\{\{\mathsf{which}_0, \mathsf{mango}_1\}, \{\mathsf{C}_2, \{\mathsf{ripened}_3, \{\mathsf{which}_0, \mathsf{mango}_1\}\}\}\}$$

(c) Sets

Figure 1: Structures, both with and without explicit features

ternative theories, but rather alternative modes of presentation of the same underlying truth. Thus at any point the one should be used which is best suited for the expository task at hand.

While an analysis is completely determined by a lexicon, a set of lexical items with feature bundles, we will present our analyses rather in terms of *lexical flowcharts*, as in figure 2, which are more visually accessible. From such a flowchart, we can exactly reconstruct a feature bundle filled lexicon, but the flowchart allows one to easily visually trace out the possible deriva-

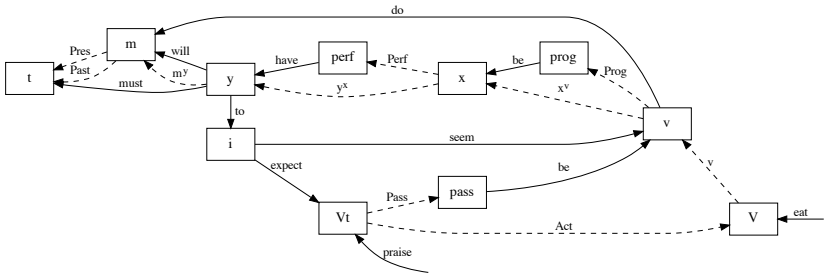


Figure 2: A lexical flowchart

tions implicit in the lexicon.

The main contribution of this paper is the introduction of a novel analysis method, which I call *(lexical) decomposition*. Lexical decomposition is made possible by the fact that the Minimalist structure building operations (Merge, both internal, external, and head) are rich enough to permit lexical rules to be implemented as lexical items. This provides for a way to eliminate redundancy from a given lexicon, effectively yielding a mechanism for updating a grammar so as to optimize an evaluation metric [Chomsky, 1965]. Lexical decomposition yields a (partly) *mechanical* method for optimizing an analysis, which converges to familiar linguistic ones. We can thus see that certain well-known analyses are not random points in analytical space, but are rather optimal points given very basic initial assumptions (couched in terms of whole-word dependencies).

This paper consists of two major parts. Section 1 introduces the formal framework of Minimalist grammars, which is then used in section 2 to develop a familiar analysis of the English auxiliary system Chomsky [1957], Lasnik [1995], as well as other canonical A-movement phenomena.

1 Formal foundations of Minimalist syntax

This section introduces the formal framework of Minimalist grammars, albeit from an unorthodox perspective. Instead of beginning with the syntactic operations of internal and external Merge, and a discussion about how these put expressions together, we begin with a discussion of features (in section 1.1) and feature checking (in section 1.2). Describing which features of which lexical items check which others, one ends up with a graph (what Stabler

[1999] calls a *matching graph*), whose edges connect pairs of features which check each other at a derivational step. This graph provides a simple record of derivationally established feature checking relationships. An important role of formal work is to identify the essential properties of theoretical objects. Section 1.3 establishes that these matching graphs are informationally equivalent to the more familiar syntactic structures Minimalists use to represent the structure of expressions, as well as dependency structures of a certain kind. Section 1.4 discusses various issues concerning the interface to PF.

1.1 Features and Feature Bundles

One basic idea is that properties of lexical items drive the syntactic computation [Chomsky, 1993]. These properties are reified into what we will call *features*. I will assume there to be only a finite number of basic syntactic features in a given language - of syntactically relevant properties of lexical items. There is no claim being made about whether they are brute or emergent, or whether they are uniform across languages. Minimalist syntacticians tend to view syntactic features as (at least partly) reducible to morphological features. This reductive stance is perfectly compatible with the formal setup here, as would be its opposite. Adopting the more general perspective as I do here fits better with the goal of formalizing the *program*, not a particular one of its instances.

Features are the syntactically relevant properties of lexical items, but what exactly are these? Borrowing terminology from the dependency literature, in a complete sentence, a lexical item will govern certain others, and will be a dependent of (or be governed by) still others. In constituency terminology, we might rephrase this as saying that a lexical item will contain other (maximal projections of) lexical items in its maximal projection, and will itself be contained in the maximal projections of other lexical items. This is commonly talked about in terms of *selection*, with the expression in a lexical item's maximal projection being those it selects, and it appearing in those maximal projections which select it. These are two very different things, and accordingly a lexical item will have two kinds of features: one kind (a *positive* feature) which is relevant for determining which lexical items it will govern, and another kind (a *negative* feature) which is relevant for determining which lexical items may in turn govern it. For any feature x, its positive version will be written as $\mathtt{x}^{+}$, and its negative one as $\mathtt{x}^{-}$. This identifies *selection* with the property of lexical items which drives the syntactic computation.

Although I assume there to be only a finite number of basic feature types in a given langauge, a lexical item might have multiple of them. A collection of features will be called a *feature bundle*. When a lexical item has multiple features, an important mechanical question is which feature is to be used next? The *locus principle* (attributed by Collins [2002] to Chomsky [2000]) is one such, stating that (in our terms) no negative feature may be used if there are positive features available. There is a tendency in the literature to find extrinsic principles which determine the order in which features are to be used (which may then vary across languages, as in Assmann et al. [2015]). Despite the empirical and theoretical differences the various approaches in the literature may have, the end result is often the same: features in a bundle are used in a particular order. I adopt this here, but simply stipulate that feature bundles are structured as a finite list of features, remaining agnostic about whether this should be derived from something more basic. It is at any rate very convenient to be able to write down a feature bundle as a sequence of features (as opposed to a partially ordered multi-set, for example).

1.2 Feature Checking

The point of positive features is to indicate a need to *govern* something, and the point of negative features is to indicate a need to *be governed* by something. As an example, consider a wh-determiner like `which`. `Which` requires an NP complement, itself forms a DP, and requires being licensed as a *wh*-word. The syntactic features it has then can be represented as follows. It needs to select an NP, so it has a feature $\mathtt{n}^+$. It can be selected as a DP, so it has a feature $\mathtt{d}^-$. And it must be further licensed as a wh-word, so it has a feature $\mathtt{w}^-$. Moreover, these features must be satisfied in a particular order. We would like words to satisfy their selectional properties before themselves being selected for, and so in its feature bundle the positive feature ($\mathtt{n}^+$) must precede the others. General syntactic considerations dictate that it should be selected for before being *wh*-licensed, and so in its feature bundle the negative feature $\mathtt{d}^-$ should precede the negative feature w. Thus, the lexical entry for the word `which` could look as follows:

$$\mathsf{which} :: \mathtt{n}^+.\mathtt{d}^-.\mathtt{w}^-$$

This lexical entry contains two kinds of information. The first, written to the left of the colon, specifies the name of the lexical item (in this case, which).[2]

[2]The name of a lexical item is important for determining how it is interpreted at the interfaces, although it is often convenient to conflate the name of a lexical item with some

The second, written to the right of the colon, specifies its syntactic feature bundle.

The distinction between positive and negative (or governer and governee) features mirrors a distinction between internal and external syntactic structure. The internal structure of a word is everything in its maximal projection, and this is determined by its positive features. The external structure of a lexical item refers to where it's maximal projection occurs within a larger structure, which is determined by its negative features. We first concentrate on the internal syntax of which. Its feature bundle indicates that it requires a noun phrase. A lexical item like mango is a NP in and of itself, and so it bears the (negative) feature $\mathtt{n}^-$. It neither requires any syntactic arguments, nor does it participate in any further syntactic dependencies. Thus the lexical entry for the word mango could look as follows:

$$\mathsf{mango} :: \mathtt{n}^-$$

These lexical items (which and mango) 'fit' together in a syntactically relevant way: the first feature in the feature bundles of both are the positive and negative version respectively of the same atomic feature. In this situation, the two features *cancel each other out* (or can be *checked*). We can represent this by drawing a line between these two features, as shown in figure 3.[3]

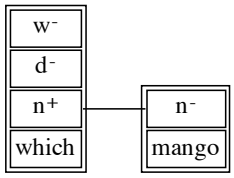


Figure 3: Checking features

After doing this, we have linked together two lexical items. This creates

aspect of its pronunciation. This practice breaks down in the face of certain kinds of homonymy, and becomes very unwieldy the farther removed from wordhood the lexical item is.

[3]To make this visualization more straightforward I have written the feature bundles vertically, from bottom to top instead of from left to right, and atop the lexical item's name,

something (a connected graph) which I will call a complex syntactic object. The complex syntactic object so connected has a *head*.

Definition (Head)**.** *A head is an item in a connected graph without any checked negative features.*

I assume that every lexical item has at least one negative feature, which we can understand as its part of speech, or *category*. Note that a lexical item is its own head, under this definition. A complex syntactic object may also have unchecked features in the feature bundles in different parts of this complex. In the case of figure 3, there is only one feature bundle with unchecked features, and these are the $\mathtt{d}^-$ and $\mathtt{w}^-$ of which. Note that which also has no checked negative features; it is therefore the head of this syntactic object.

As there are no more unchecked positive features, we are done with the internal syntax of which. As which mango is a DP, it must be selected by something which subcategorizes for a DP. As an example, we may choose the verb `ripened`. This is a tensed intransitive verb, which means that it selects a DP argument ($\mathtt{d}^+$), and then may be selected as a tensed phrase ($\mathtt{t}^-$). Its lexical entry might look as follows:

$$\mathsf{ripened} :: \mathtt{d}^+.\mathtt{t}^-$$

While the first feature of the feature bundle at the head of the syntactic complex which mango (the checked $\mathtt{n}^+$) does not fit together with the first feature of ripened (the $\mathtt{d}^+$), the first *unchecked* feature of which mango (the $\mathtt{d}^-$) does. I stipulate that only the first unchecked feature of a feature bundle is accessible for checking at any time.

Stipulation 1 (Accessibility)**.** *Only the first unchecked feature in a feature bundle may participate in checking*

Thus feature checking makes new features accessible in a feature bundle. With this in mind, the lexical item ripened fits together with the complex which mango as their first unchecked features are identical but of opposite polarity. Note crucially that ripened does **not** fit together in this way with the *lexical item* which by itself. We can connect the matching first unchecked features of both feature bundles with a line, creating a larger syntactic complex, the head of which is now ripened (now that which has had its $\mathtt{d}^-$ feature checked, the only lexical item without any unchecked features is indeed ripened). Note that this larger complex expression has *two* feature bundles with unchecked features; one (attached to which) contains just the unchecked

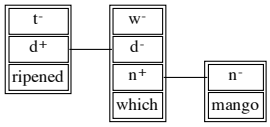


Figure 4: adding the verb

feature $\mathtt{w}^-$, and the other (on the head, ripened) contains just the unchecked feature $\mathtt{t}^-$.

In our semi-naïve background syntactic theory, we need a *wh*-licensor for the *wh*-feature on the *wh*-word. This is usually thought to reside in a head above tense. Adapting this idea into our formal theory, we must have a new lexical item which selects a tensed phrase $\mathtt{t}^+$, and then which licenses a *wh*-phrase ($\mathtt{w}^+$). Bowing to tradition, we may call the category of this head a complementizer ($\mathtt{c}^-$). As it contributes nothing to the pronunciation of the sentence, it is not useful to give it a name reminiscient of its pronunciation, and so we will name it C.

$$\mathsf{C} :: \mathtt{t}^+.\mathtt{w}^+.\mathtt{c}^-$$

Our rules for connecting expressions force us to **first** draw a line between the feature $\mathtt{t}^-$ in our complex expression ripened which mango and the $\mathtt{t}^+$ in the lexical item C. This is because the $\mathtt{t}^+$ is the first unchecked feature in the feature bundle of C. This constructs a new complex expression containing all four lexical items, with two not-completely checked feature bundles: $\mathtt{w}^-$ and $\mathtt{w}^+.\mathtt{c}^-$.

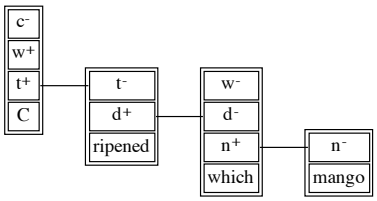


Figure 5: adding a complementizer

In this expression, the only node without any checked negative features is C. While which has an unchecked negative feature $\mathtt{w}^-$, it also has a checked

d^- feature. Therefore, C is the head of this expression.

For the first time we have a complex expression which contains two feature bundles with matching first features, thus satisfying in itself our criterion of 'fitting together'. We thus draw a line connecting the postive and negative versions of the w feature. In the resulting expression, there is but a single feature which is unchecked, and it is the category feature of the head. We call such an expression *complete*.

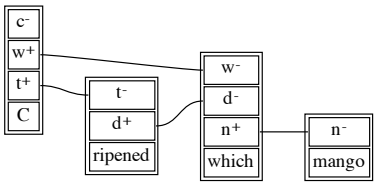


Figure 6: checking the wh-features

We have ended up with a graph indicating feature checking relationships between lexical items. To define such a graph, one simply connects matching unchecked features to one another one after the other. This process can be thought of as a derivation, and the lexical items making up the graph the *numeration*. The entire derivation is given again in figure 7. Moving to numerations, we need to be more careful with how we understand our previous (and upcoming) definitions and stipulations regarding feature checking. The definition of *head* given previously states that a head is an item in a connected graph without any checked negative features. Once we have a numeration, our graph consists of multiple disjoint connected subgraphs. We must therefore understand these definitions and stipulations as speaking of connected *subgraphs*. For example, the initial numeration in figure 7 consists of four maximal (trivially) connected subgraphs: each node is a maximal subgraph, and is therefore a head. After the n features of which and mango are connected together, there are three maximally connected subgraphs: C and ripened continue to be their own connected subgraphs (and therefore their own heads), but now the two node graph consisting of which and mango is a maximally connected subgraph, it's head is of course which.

It is worth noting that the basic operation of adding an edge between two lexical items is strictly monotonic — it exclusively adds to a representation, and does not otherwise modify it; this is the content of the *No Tampering Condition*. Squinting, we can identify adding an edge with the operation

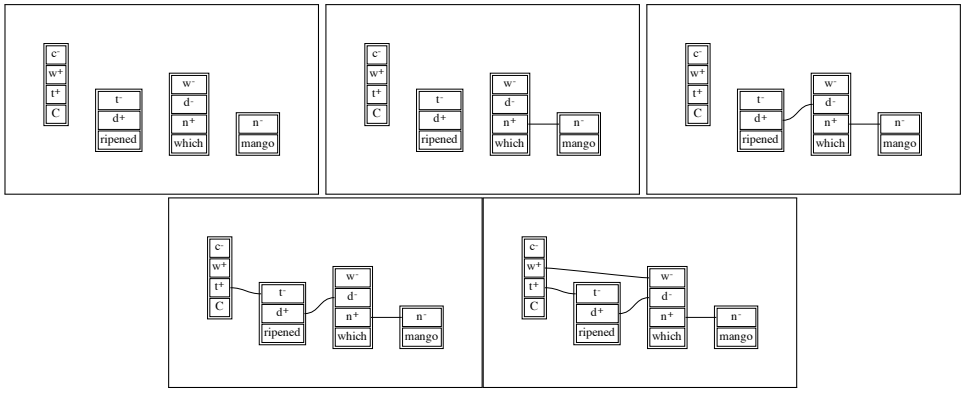


Figure 7: Constructing a feature checking graph for "Which mango ripened"

Merge, and we see that, indeed, internal and external Merge are identical operations. "Note that both operations come free: it would require stipulation to bar either of them. Furthermore, there are no operations 'form copy' or 'remerge,' just simple Merge." [Chomsky, 2013] Although we have just been recording which features of which lexical items check each other, there appears to be some tantalizing connection between this and the more standard presentations of Minimalism. We will see shortly (in section 1.3) that this is not an accident.

Our rule for constructing edges, "connect the first unchecked features of matching feature bundles," has some undesired corner cases. As it is stated, nothing stops us from taking two syntactic complexes, and checking features from feature bundles *which are not at the heads* of the complexes; this gives rise to *grafting*, *parallel merge*, and *sidewards movement* [Nunes, 2001, Citko, 2005, van Riemsdijk, 2006].[4] I wish to block this in this chapter. I will stipulate the following condition on feature checking:

Stipulation 2. *The head of a syntactic complex must be involved in any feature checking relationship*

If there are two syntactic complexes, then both their heads must be involved (i.e. must contain one of the pair of matching features). However, if there is but one syntactic complex (as there was in the last step of our previous derivation, where the w features were checked), its head must be involved, but this allows a non-head feature bundle to host the other feature.

[4] As these authors note, it requires stipulation to block any of them, given Chomsky's published characterizations of Merge.

Another fringe case involves *competition* between features. As an example, if there had been multiple feature bundles in our previous example with an accessible $\mathtt{w}^-$ feature, any one of them could have checked the $\mathtt{w}^+$ feature at the head. I will want to prohibit this as well.[5]

Stipulation 3 (SMC)**.** *A feature may check another only if there is no other accessible feature which could have checked it*

The proper way of understanding this stipulation when working with a numeration is to consider just the maximal subgraphs which contain the two features in question. Thus this restriction also prohibits us from using a feature from a separate complex to check a feature that could have been checked from within the complex (in now outdated parlance, it enforces a **move over merge** constraint [Shima, 2000, Wilder and Gärtner, 1997]). As an example, if we had a lexical item why :: $\mathtt{w}^-$ with just a $\mathtt{w}^-$ feature in its feature bundle, we could not have used this feature to check the $\mathtt{w}^+$ feature of our ripened which mango complex, because there is *already* another accessible feature which could check it, namely the $\mathtt{w}^-$ feature in the bundle of which. This constraint does not stop us from using the $\mathtt{d}^-$ feature on which mango to check the first accessible $\mathtt{d}^+$ feature of devoured, even if John (which contains its own $\mathtt{d}^-$ feature) is elsewhere in the numeration. This is because the two maximal connected subgraphs in question, which mango and devoured, do not have a competing feature between the two of them. We could of course have chosen in this same setting to use the $\mathtt{d}^-$ feature of John instead; the structure of the numeration underdetermines the feature checking relationships ultimately arrived at.

Our insistence on feature checking has given pride of place to lexical feature bundles. This is faithful to early presentations of Minimalism [Chomsky, 1995], but deviates from more recent approaches [Chomsky, 2004] in which Merge applies freely. It is easy to imagine converting the present system into one where edges are freely drawn between lexical items. (Simply ignore the features in our example, and draw edges willy-nilly.) In a free Merge system, the role of filtering out ill-formed structures falls exclusively on the interfaces. As discussed in Kobele [2014] and Graf [2017], the filtering behaviour of simple interfaces (those which satisfy a condition called 'regularity') can be encoded as features on lexical items in exactly the way done here. In the same way, the feature checking requirement on drawing edges can be pushed

[5] This constraint, called the SMC in Stabler [1997], ensures that the typological predictions of Minimalist grammars are responsible. See Graf's contribution to this volume for more details, especially regarding relaxations of and alternatives to this constraint.

out of core syntax and into the interfaces. In the second part of this paper, we will do something quite similar; we will begin with the desired graphs, and will extrapolate from them feature bundles that license their construction.

Our description so far has not touched on things like word order (how do we pronounce a complex syntactic object), or derived syntactic structure (how can we represent the structure of a complex syntactic object in a familiar way). Instead, all we have done so far is to indicate which features have checked which others. Presenting syntax in this way disentangles the specification of dependencies from structure building; two logically distinct things. Establishing dependencies via feature checking is at the core of our syntactic formalism, and this is logically distinct from what derived structure we decide to assign, or how we decide to linearize it. We will move on to these questions soon enough (and they will be easy to answer, now that we have this feature checking idea firmly in place).

1.3 Structure Building

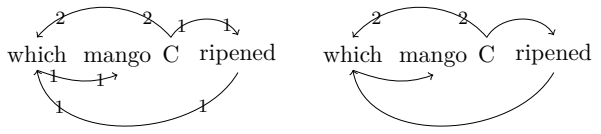
In the previous section, we saw how features could be used to control the addition of edges to our graphs. In this section, I turn to the topic of syntactic structure building, and show that it can be thought of as a reflex of edge construction. One reason for distinguishing between feature checking (or edge construction) and structure building is that assumptions about derived syntactic structure are largely orthogonal to the relationships between lexical items that undergird them.

I will begin by identifying the meaningful information currently recorded in our matching graph representations from the previous section, and considering how that information might be presented in different ways. The nodes of our graphs have not been atomic, but have instead been structured objects in their own right (towers with features on each level, and on the bottom a lexeme). This additional structure in the nodes was used to specify which of a lexical item's features checked which of another's features. This is the fundamental information that needs to be conveyed. Instead of representing this information on the nodes of the structure, we can instead encode this information elsewhere. In addition to recording which features in the two feature bundles check each other, we must also find a way to represent the asymmetry between positive and negative features. It is convenient to encode this information onto the edges themselves. Edges are directed, *from* the lexeme with the positive feature *to* the lexeme with the negative feature. Three different choices about how to encode which features of lexical items check each other are presented below.

While it is natural to want to ask, given three representational alternatives, which is the *correct* one, that misses the point here.[6] All of the following representations are equivalent to each other, in that they depict the same underlying structure, much as polar and cartesian coordinates allow for describing the same points on the plane. If we were to have only a single coordinate system, it would be easy to confuse properties of this system with properties of the points they represent. Having multiple nominally different representations for the objects we are interested in allows us to peer beneath the luster of our notation, and catch a glimpse of the reality it clothes. Unlike the polar and cartesian coordinate systems however, where one representation can prove at times drastically more concise than another, the representational schemes below (with the exception of the set-theoretic one, which is strictly dominated by the others when movement is involved) are equally concise (up to a constant factor).

1.3.1 Dependency structures

One option is to simply label each edge with the position in the two feature bundles of the features it checks. The edges are labeled with pairs of numbers, indicating the position of the positive feature in the first, and the position of the negative feature in the second feature bundle. The number indicating the position of the positive feature is placed near the edge's tail, and the number indicating the position of the negative feature near its head. This is depicted in figure 8(a). Impressionistically, most dependencies involve either a first positive feature or a first negative feature. It is therefore convenient to leave the number '1' on an edge implicit; the same structure with this convention is depicted in figure 8(b).



(a) All numbers (b) Implicit ones

Figure 8: Indicating connections between features via numbers on dependencies

Focussing on the word `which` in figure 8(a), we see that there are two

[6] Indeed, it is not at all clear that this question is even meaningful in this case, unless one makes extremely strong assumptions about the nature of mental representations.

incoming edges, the first from ripened, and the second from C, and one outgoing edge (to mango). That its outgoing edge is the first incoming edge to mango tells us that the first positive feature of which ($\mathtt{n}^+$) checked the first negative feature of mango ($\mathtt{n}^-$). That its first incoming edge is the first outgoing edge of ripened tells us that its first negative feature ($\mathtt{d}^-$) was checked by the first positive feature of ripened ($\mathtt{d}^+$). And that its second incoming edge is the second outgoing edge of C tells us that its second negative feature ($\mathtt{w}^-$) was checked by the second positive feature of C ($\mathtt{w}^+$). The head of the expression is identifiable as the node with no incoming edges, in this case C.

In this representational scheme, Merge amounts to adding a dependency between two lexical items (with an appropriate label).

Osborne et al. [2011] note a similarity between Minimalism and dependency grammar. The structures we presented here are dependency structures, very close to those used in Word Grammar [Hudson, 2010], which allow a single node to have multiple incoming edges (multiple dominance); precisely what is demanded when movement is in the picture. What distinguishes these dependency structures from those of Word Grammar (or other traditions) is the content of the labels on edges. In Word Grammar, these labels indicate grammatical relations (such as *subject*, *modifier*, or *object*), whereas here they are indicate which pair of features the edge connects. If certain formal assumptions on feature bundles are imposed, such as a prohibition on having multiple identical features in one feature bundle, then feature names can replace pairs of numbers as the labels on edges. In the context of further common assumptions, we might imagine that there could be a correspondance between feature names and grammatical relations; for example, a case feature could be thought of as a generic *argument* relation, while a topicalization feature a *TOPIC* operation, etc. This would bring the formal difference between Word Grammar and Minimalism to a minimum, making Minimalism seem like a derivational approach to Word Grammar.[7]

[7] A substantive (as opposed to a formal) difference is that Word Grammar makes use of a richer set of dependency labels, such as *subject* and *object*, whereas Minimalism deliberately attempts to decompose these into the label I have called here *argument*, which is interpreted as either *subject* or *object* depending on the lexical item to which it points: pointing to an AgrS means it is a subject, to an AgrO an object. Similarly, Minimalism makes use of many more covert words than does Word Grammar. Lexical decomposition as discussed in section 2.1.1 allows us to make sense of this difference — the covert words of Minimalism are exactly those which emerge from the whole words of Word Grammar by the lexicon optimization process of lexical decomposition.

1.3.2 Mirror theory

The information contained in matching graphs can be presented in other ways as well. The following representation makes heavy use of node layout: vertical to indicate which of two nodes contained the negative, and which the positive feature, and horizontal to indicate which feature was checked. Given a node m, it is depicted as being above all other nodes which its positive features are checked by, and these 'daughters' are ordered from left to right as per which of this node's features they checked. The incoming edges to a node are also ordered from left to right to indicate which feature each edge links to which other. This is represented in figure 9.

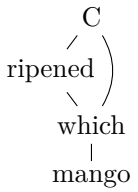


Figure 9: A layout-based representation

Focussing again on the word which in figure 9, it has one daughter (i.e. one outgoing edge), mango, indicating that its first (and only) positive feature is checked by a negative feature of mango. The word which has two parents (two incoming edges), the leftmost one, ripened, accordingly checks its first negative feature, and the second one, C, its second negative feature.

In this representational scheme, Merge amounts to adding an edge between nodes.

This representation is at the heart of Brody's *Mirror Theory* [Brody, 2000], which represents in addition whether an edge is a morphological word building dependency, co-opting linear order to encode this information, and imposing an upper bound of two on the number of outgoing edges a node may have (which amounts to placing an upper bound of two on the number of positive features in any lexical feature bundle).

1.3.3 Bare phrase structure

Our third representation mixes the original 'complex node' matching graph representation with the layout based one from figure 9. The individual positive features of a word are represented as independent nodes, which I label with a circle '$\circ$' for convenience. Their linear order in the feature bundle is

represented via the dominance relation. By virtue of splitting features off of nodes, this representation is able to restrict the number of outgoing edges of any node to a maximum of two. The number of positive features on each lexical item allow us to reconstruct which circles belong to which words in this structure, thus freeing up linear order to be used for other purposes.

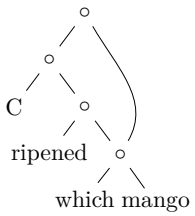


Figure 10: Positive features represented as nodes, order via dominance

Of course, this structure is nothing more than the kind of constituency structures familiar in modern transformational syntax, with chains represented via multiple dominance.[8] Each node beyond the first in the projection of a lexical item 'hosts' exactly one of its positive features. This constituency structure was already implicit in the dependency structure from the previous section. All we did was rearrange the information contained in the dependency structure. Whereas the dependency representation forces us to keep track of the *order* of the dependencies a lexical item's positive features enter into, this order information is encoded in a constituency tree via the dominance relation. Note that, of course, we can go from a multiple dominance structure of this sort back to a matching graph by 'squishing together' the nodes projecting from the same head. We need to remember to encode the immediate dominance relations among these nodes as precedence relations in the lexical feature bundles.

In this representational scheme, Merge amounts to adding a new root node, and two edges from this node to the roots of the structures merged.

Finally, if we assume that each time a lexical item is selected from the lexicon it receives a unique index, we can encode the phrase structure tree in figure 10 in the set theoretic way recommended by Chomsky [2000], and (following Collins [2002]) without labels. Each internal node is interpreted as an instruction to form the set of its two daughters. This gives us the set

[8]The foundational reference to multi-dominant syntax in Minimalism is Gärtner [2002]. Kracht [2001] compares the formal properties (and intertranslatibility) of copies, multiple dominance, and traces as representations of the derived structures of movement.

theoretic object below:

$$\{\{\mathsf{C}_i, \{\mathsf{ripened}_j, \{\mathsf{which}_k, \mathsf{mango}_l\}\}\}, \{\mathsf{which}_k, \mathsf{mango}_l\}\}$$

Relating this back to the previous section, where matching graphs were first introduced, we see that the edges connecting matching feature pairs indeed represent Merge steps, and can in fact be viewed as constructing sets.

1.4 Interfaces: Linearization

In the last section I tried to show that the matching graphs we obtain by linking lexical features together just *are* the tree-like structures with movement we are familiar with, in disguise. Thinking of representations in terms of the information they encode provides us with a different perspective on how to understand questions about and arguments for one or the other representation. In addition, different but informationally equivalent representations can suggest different but equally valid perspectives on grammaticality; the matching graph emphasizes that grammaticality can be viewed as verifying that features of lexical items were checked off against each other in the right way.

Of course, these abstract structures have very little contact with the rich empirical data that we would like to account for; they account for perhaps the number and identity of words (i.e. lexical items) in sentences. While this is already of some interest, it is not of ***that much*** interest: *on alligator of anaconda the road the side the ate* and *the anaconda ate the alligator on the side of the road* are identical w.r.t. the numerosity of their words, but one is word salad, the other an everyday occurrence in Florida. Much more interesting would be if we could account for not only the numerosity, but also the linear order of words in sentences.[9]

As we have shown in the previous section how to move from dependency structures to (multiple-dominance) trees and back, we can describe how to linearize our structures by specifying how to linearize (multiple-dominance) trees. There have been many suggestions as to how to spell out (linearize) an unordered multiple dominance structure, or equivalently, a tree with multiple copies. It is often convenient to attempt to solve new problems by adapting solutions to old problems. As we know how to linearize ordered trees, we can try to construct ordered trees from an unordered MDS. To do this, we must solve two independent problems.

[9] We would also like to account for some aspects of sentence meaning, but that is too advanced for the present chapter. Kobele [2006, 2012] shows how to apply the Heim and Kratzer [1998] approach to semantics to Minimalism in a directly compositional way.

1. turn an unordered MDS into an unordered tree

   essentially: choosing which chain link to pronounce

2. turn an unordered tree into an ordered tree

   essentially: how to order sisters

There are many ways of solving these two problems, some of which correspond to proposals in the linguistic literature. I will begin with what I take to be the simplest popular answers to these questions, before generalizing these answers and showing how they can be visualized in terms of properties of lexical feature bundles.

1. each multiply dominated expression is to be exclusively pronounced in its structurally *highest* position

2. specifiers are pronounced before heads which are pronounced before complements

1.4.1 Ordering

The simple answer to the ordering problem (SPEC-HEAD-COMP) treats every lexical item (in every language) the same; you are pronounced to the left of the expression that checks your first positive feature, and to the right of all expressions that check your other positive features. Furthermore, an expression checking a later positive feature of yours will be pronounced before (i.e. to the left of) an expression checking an earlier positive feature of yours.

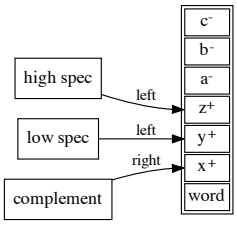


Figure 11: The lexical item word :: $\mathtt{x}^+.\mathtt{y}^+.\mathtt{z}^+.\mathtt{a}^-.\mathtt{b}^-.\mathtt{c}^-$, and the structural and word-order properties of its governees

It is common to analyze German as having "complement-head" order in the verbal and tense domains, but "head-complement" order elsewhere. This 'mixed' word order property of German, if we accept it at face value, means that we cannot hold onto this simple answer to the ordering problem.[10] Instead, German word order properties are naturally stated by reference to particular lexical items (verbs and auxiliaries require Comp-Head order, others Head-Comp order). We could implement this lexical item-dependent word order variation at the featural level, by specifying for each positive feature on each lexical item, whether its checker should appear to the left or right of the head.

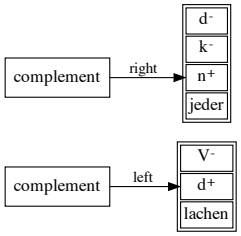


Figure 12: Feature-based word order variation in German

This information needs to be recorded somehow. It is simple to attach linearization instructions to the individual features. We can write $^{+}\mathtt{x}$ for a positive feature which wants its checker to be on its left, and $\mathtt{x}^{+}$ for a positive feature which wants its checker to be on its right (so the plus is on the side that the dependent should be linearized on). With this convention, we can write lexical items for German words that disagree about what side their complements should appear on:

Table 1: Putting linearization under lexical (featural) control

Head-Comp	Comp-Head
jeder :: $\mathtt{n}^{+}.\mathtt{d}^{-}.\mathtt{k}^{-}$	lachen :: $^{+}\mathtt{d}.\mathtt{V}^{-}$

[10] Of course, we **could** keep this simple answer if we made our analysis more complicated, taking at least one of the two word orders to be the result of additional movements.

1.4.2 Chains

The simple answer to the multidominance problem[11] again treats every lexical item the same; it and its maximal projection are pronounced in the position checking their last feature. Because all dependencies entered into by lexical items are determined by their feature bundles, we can view this pronunciation scheme in terms of the features in a lexical item's feature bundle, as depicted in figure 13.

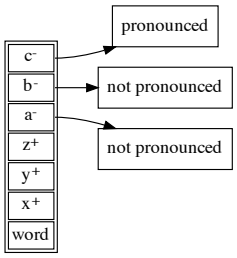


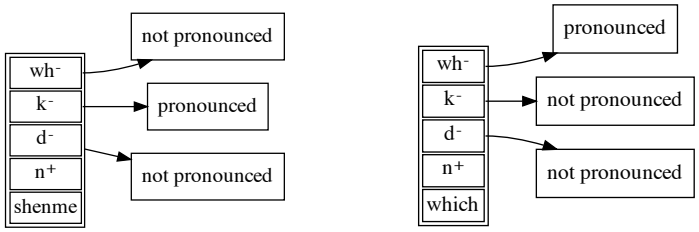
Figure 13: The lexical item word :: $\mathtt{x}^+.\mathtt{y}^+.\mathtt{z}^+.\mathtt{a}^-.\mathtt{b}^-.\mathtt{c}^-$, and its pronunciation

There are two obvious ways of generalizing this. The first way is parallel to the way we allowed each lexical item to determine for itself the pronunciation orders of its dependents above. This amounts to allowing a lexical item to specify which of its features it will be pronounced in. Figure 14 presents a simplified picture on the difference between *wh*-movement and *wh*-in-situ languages, where both *wh*-strategies are viewed as syntactically identical, with the difference residing solely in the choice of chain link to pronunce.

A second generalization of copy spellout takes control away from the lexical item itself, and moves it to the interaction of its governing heads. This seems most reminiscient of actual practice, where a governing head which can support pronunciation is said to have an **EPP** feature. The intuition underlying this generalization is that dependencies able to support pronunciation *pull* a maximal projection up from its default base position.[12] Pursuing this

[11] This problem is not particular to the multidominance representation. It appears in exactly the same way with multiple coindexed copies.

[12] A particularly fine-grained version of this idea comes in the form of complexity filters [Koopman, 2014], where each position may place restrictions on not only *whether* it can host an overt expression but also on the syntactic and prosodic shape of this expression. Kobele [2011] investigates conditions under which the addition of complexity filters is



(a) Chinese (*wh*-in-situ) (b) English (*wh*-movement)

Figure 14: Cross-linguistic variation in pronunciation height in English and Chinese

metaphor, a dependency is called **strong** just in case it supports pronunciation (i.e. is able to pull up a maximal projection), and is **weak** otherwise. In contrast to our previous generalizations, which were statable in terms of the properties of a particular lexical item in isolation, we cannot predict on the basis of a particular lexical item in which of its negative feature positions it will ultimately be pronounced - its ultimate pronunciation position is only determinable on the basis of properties of the structure in which it finds itself. When multiple positions compete for hosting the pronounced version of the phrase, we will say that the (structurally) highest one wins (that is, the one which checks the latest feature). If **no** position is able to host the phrase, it is pronounced in its first negative position.

We can indicate on a feature whether it is able to support pronunciation or not by adding a tilde above the name of a feature which does *not* support pronunciation - in other words, weak features are marked as such with tildes. This is shown in table 2.

Table 2: 'EPP' notation

Weak	Strong
$\tilde{\mathtt{x}}^{+}$, ${}^{+}\tilde{\mathtt{x}}$, $\tilde{\mathtt{x}}^{-}$	$\mathtt{x}^{+}$, ${}^{+}\mathtt{x}$, $\mathtt{x}^{-}$

Our first generalization (where each lexical item decides for itself which of its negative features will host its pronunciation) amounts to allowing only negative features to be weak. The last strong feature in a feature bundle is

admissible in the Minimalist grammar formalism.

the one where the lexical item's maximal projection will be pronounced. If all features are weak, it will be pronounced in the first one. This generalization would be able to capture a situation where a particular position was able to host expressions of a certain type, but where some expressions of that type were always pronounced lower than that position.

Our second generalization (where the ultimate pronunciation site of a maximal projection is a global property of the structure) amounts to allowing only positive features to be weak. The maximal projection of a lexical item will be pronounced at the position of the last of its features checked by a strong feature, or in the position of its first negative feature if all of its negative features are checked by weak ones. This generalization would be able to capture a situation where some words allow overt movement to their specifiers, but other words require the same expressions to move covertly. Some people think of case in English as working in this way. Clearly, all DPs in English **can** be pronounced in the position which assigns them case (esp. nominative case). However, many people analyze objects as being in their in-situ positions, despite being assigned case by a higher head. This could be accounted for by assigning a strong positive case feature to the T head, and a weak positive case feature to (say) the v head.

Of course, we could combine these generalizations, allowing strength and weakness to be assigned *ad libitum* in feature bundles. The strong/weak ***negative*** features in a lexical item's feature bundle indicate where a lexical item is able to be pronounced. The strong/weak ***positive*** features in a lexical item's feature bundle indicate where a lexical item is able to overtly host something. Then the maximal projection of a lexical item will be pronounced in the highest of its strong negative features which is checked by a strong positive feature. The question of what to do in case none of a lexical item's strong negative features are checked by strong positive features is left open by this formulation, but could be naturally resolved either by causing such a derivation to crash (at the interface), or by defaulting to some designated position (say, the first negative feature, regardless of strength).

Many other pronunciation schemes are possible. The interest of ones based on strong/weak features is that they are *lexicalizable* - they can be formulated in terms of lexical feature bundles. As discussed in section 1.2, features can be thought of describing the filtering effects of regular interface maps on freely Merged structures. From that perspective, lexicalizable pronunciation schemes reflect *local* (as opposed to global) interface computations.

1.4.3 Informational equivalence

As the matching graphs over feature towers (and numbered dependency structures) are informationally equivalent to multiple dominance structures, one should be able to linearize these structures directly. Indeed we can, and in a simple and straightforward way. The twin problems of *ordering* and *chains* emerge in this context again, with exactly the same character. Indeed, the proposals in sections 1.4.1 and 1.4.2 are representation agnostic; they are stated in a way that is independent of the representation chosen, and the ideas therein can be applied without change to these other structures.

When linearizing dependency structures, one must say something about how to determine the relative order of the immediate dependents of a parent. This requires the postulation of some asymmetry between the dependents, to provide a scaffolding for the desired asymmetry in linear order. This issue does not arise in the context of (binary branching) constituency structures, as the constituency of the structure itself imposes such an asymmetry on dependents. The structures used here, the matching graphs over feature towers and the numbered dependency structures, also impose an inherent order on dependents. In the case of the matching graphs over feature towers, it is the position of the features in the tower, and in the case of the numbered dependency structues, it is the number of the dependencies. Both of these (and also the constituency of the constituency structures) ultimately reduce to the order of features in lexical feature bundles.

2 Decomposition as a discovery procedure

The previous sections have introduced a formalism in which we may write Minimalist analyses. The formalism was presented as a system for constructing links between lexical items. The resulting matching graphs over feature towers, while initially unfamiliar, are actually equivalent to the structures involving movement that we normally use. While links between lexical items could in principle be drawn in many ways (e.g. 'if your graph has an even number of edges, remove half of them at random, otherwise make two additional copies of each edge and add a new edge at random'), the system we have been exploring *lexicalizes* the link drawing procedure, by putting features on lexical items. This is a useful starting point, although we can imagine ways of relaxing this.[13] These dependency-establishing features on

[13]We might allow links to be drawn, without any syntactic features involved at all, between two items which are in a semantic function-argument relation (for example, a verb and its argument). We would have to keep track of the semantic type of expressions,

lexical items can also be exploited to deal with linearization (directionality of selection and chain link pronunciation), as detailed in the previous sections. In this section I use the system we have developed to reconstruct familiar analyses of basic phenomena. I will continue to emphasize the role which can be played by lexical feature bundles in this endeavor.

Section 2.1 begins with a simple, whole-word analysis of the English auxiliary system. While simple, this analysis misses generalizations. We see that these missed generalizations manifest themselves as redundancies in the lexicon. Section 2.1.1 introduces the formal mechanism of *decomposition*, which shall be used to eliminate lexical redundancies. Decomposition amounts to breaking a single head into two, and so we need to introduce a way of reassembling our original words from lexical items representing word parts. This is the topic of section 2.1.3, which uses *spanning* Brody [2000], Williams [2003], Svenonius [2016] to allow syntax to construct complex words. Section 2.1.4 applies decomposition to the original whole-word analysis, resulting in a textbook Minimalist analysis.

Section 2.2 expands the scope of our analysis, introducing more tenses and modals (section 2.2.1), negation and `do` (section 2.2.2) and a discussion of *do*-support (section 2.2.3) and repair operations in general. This section introduces *lexical graphs*, which are a graphical presentation of a lexicon. Lexical graphs make information about the derivational possibilities of lexical items explicit in a way that permits easy visualization of analyses. This makes for a simple way of reasoning about, modifying, and comparing analyses.

Section 2.3 again expands the scope of the previous analysis, this time by introducing raising to subject (section 2.3.1), to object (section 2.3.2), and passivization (section 2.3.3). The analytical development proceeds again entirely by using lexical decomposition to reduce redundancies in the lexicon. Despite the myopicity of the analytical strategy, the ultimate analysis is again completely canonical.

---

and whether a particular semantic argument position has already been saturated, but this is certainly conceivable. Another option might be to allow certain links to be drawn where only one of the involved lexical items has any features. For example, a lexical item might have an *EPP* feature, and so we might draw a link between it and (perhaps even the *closest*) something else. One can imagine many more possibilities. These two have been discussed in the literature. The first under the rubric of reducing c-selection to s-selection, and the second is I think a fairly common way of understanding agreement (where a probe has unvalued features, and is searching for a goal where it can get values from).

2.1 Investigating English auxiliaries

As a simple case study, consider the English auxiliary system. Imagine a language learner (or perhaps just a linguist) being exposed to sentences like the following.

1. John eats.
2. John will eat.
3. John has eaten.
4. John will have eaten.
5. John is eating.
6. John will be eating.
7. John has been eating.
8. John will have been eating.

Assume further that the linguist has decided that the dependencies for sentence 8 are as in figure 15.

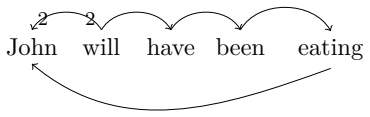


Figure 15: Dependencies for sentence 'John will have been eating'

From these dependencies, we can reconstruct the lexical items in table 3. We do this by assigning each edge a unique name, and putting the positive (resp. negative) feature with that name in the source (resp. target) lexical item's feature bundle.

Table 3: Lexical items extracted from the dependency structure in 15

John :: $\mathsf{a}^-.\mathsf{b}^-$	will :: $\mathsf{c}^+.\mathsf{b}^+.\mathsf{s}^-$
have :: $\mathsf{d}^+.\mathsf{c}^-$	been :: $\mathsf{e}^+.\mathsf{d}^-$
eating :: $\mathsf{a}^+.\mathsf{e}^-$	

After extracting lexical items from sentences 1 — 8, we have multiple 'copies' of certain words, which differ only in the names (but not the types) of features in their feature bundles. For example, there are eight (!) different copies of John, four of eating, three of will, and two each of eaten, has, and been. (This is because each time we reconstruct lexical items from a new dependency structure, we choose globally unique names for the edges.) We can unify these copies into single lexical items by renaming the features involved across the whole lexicon.[14] For example, we might decide to rename the first feature of each of the John lexical items to $\mathtt{d}^-$, which would force us to replace all features with name a with the name d, among others. After this unification procedure, we are left with the lexicon in table 4.

Table 4: Lexical items after unification of features

John :: $\mathtt{d}^-.\mathtt{k}^-$	eats :: $\mathtt{d}^+.\mathtt{k}^+.\mathtt{s}^-$
will :: $\mathtt{v}^+.\mathtt{k}^+.\mathtt{s}^-$	eat :: $\mathtt{d}^+.\mathtt{v}^-$
has :: $\mathtt{perf}^+.\mathtt{k}^+.\mathtt{s}^-$	eaten :: $\mathtt{d}^+.\mathtt{perf}^-$
is :: $\mathtt{prog}^+.\mathtt{k}^+.\mathtt{s}^-$	eating :: $\mathtt{d}^+.\mathtt{prog}^-$
have :: $\mathtt{perf}^+.\mathtt{v}^-$	be :: $\mathtt{prog}^+.\mathtt{v}^-$
been :: $\mathtt{prog}^+.\mathtt{perf}^-$	

This grammar is perfectly capable of deriving the sentences (with the appropriate dependencies) we were given originally. However, it systematically *misses* generalizations: although we know (as English speakers) that there is but a single verb, eat, which is appearing in its various forms in this lexicon, this fact is not captured in the grammar. One way to make this intuition more precise is to appeal to the *rate of growth* of our lexicon as we add more and more open class items. In order to add a new intransitive verb to our grammar we would need to add *four* separate lexical items (five, if we had included the past tense), one for each cell in its paradigm.

This is a familiar tension, and has an familiar resolution: a morphological module systematically constructs all of the lexical items needed from a single input form. This is possible because the lexical items which we need to add to the lexicon in order to capture the distribution of a novel (intransitive) verb V are not arbitrary, but have a characteristic form, shown in table 5.

Thus the redundancy in the lexicon (for each intransitive verb we have

[14]This is not innocuous, but involves a substantive hypothesis to the effect that the lexical items so unified are in fact different tokens of the same lexeme. We would presumably not, for example, want to unify bank :: i and bank :: j in the sentence "The new bank is on the bank of the river."

Table 5: Lexical items needed for an intransitive verb

V-pres :: $\mathtt{d}^+.\mathtt{k}^+.\mathtt{s}^-$	V-past :: $\mathtt{d}^+.\mathtt{k}^+.\mathtt{s}^-$
V-prog :: $\mathtt{d}^+.\mathtt{prog}^-$	V-perf :: $\mathtt{d}^+.\mathtt{perf}^-$
	V-base :: $\mathtt{d}^+.\mathtt{v}^-$

multiple lexical items) can be eliminated by postulating *lexical redundancy rules* [Chomsky, 1965, Jackendoff, 1975], here rebranded as a presyntactic morphological module. This also accounts for the productivity of the lexicon (upon learning a novel intransitive verb we add multiple lexical items).

In the remainder of this chapter we explore a different (and canonical within the Minimalist tradition) approach to the redundancy in table 4, one which exploits the structure in feature bundles.

2.1.1 Decomposition of Feature Bundles

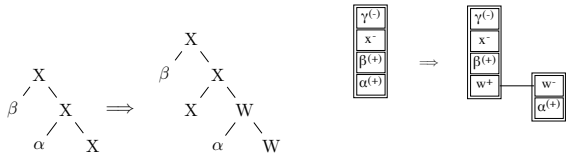
A common pattern in science is to identify *commonalities* across phenomena, factoring out a description of the phenomena into a single theory accounting for the commonalities, together with theories accounting for the individual differences. *Quod inferius est sicut quod est superius.* We can identify common patterns in lexical items, and then factor out these common patterns into new lexical items. This will allow us to express generalizations about a redundant lexicon *in the language of syntax.*

Given a lexical entry word :: α, we can divide α up into the following parts:

- the part that occurs *before* the first negative feature
- the first negative feature itself
- the part that occurs *after* the first negative feature

In a useful lexical item (i.e. one which can be used in a convergent derivation), the part after the first negative feature will consist exclusively of some number of further negative features.

So let $\alpha\beta.\mathtt{x}^-.\gamma$ be a lexical feature bundle with $\alpha\beta$ its precategorial part (one or both of α and β may be empty). We can *decompose* this feature bundle into the following two (I assume that the feature w is *fresh*; i.e. no other lexical item has a feature of that type): $\alpha.\mathtt{w}^-$ and $\mathtt{w}^+.\beta.\mathtt{x}^-.\gamma$. That is, we split the feature bundle between the α and the β.



(a) X-bar trees (b) Matching graphs

Figure 16: Visualizing feature bundle decomposition

$$\mathsf{w} :: \alpha\beta \mapsto \begin{cases} \mathsf{u} :: \alpha.\mathtt{x}^- \\ \mathsf{v} :: \underline{\mathtt{x}^+}.\beta \\ \mathsf{u} \oplus \mathsf{v} = w \end{cases}$$

Figure 17: Morphemic decomposition

From a linguistic perspective, this amounts to saying that what we used to think of as an XP (remember that `word` is of category x) is really *two* phrases: XP with complement WP. We can view this process in terms of the structures we would construct in figure 16.

As can be seen in the figures, the arguments of the original feature bundle are now distributed across two feature bundles. We want, however, to factor out redundancy not only *within* feature bundles, but within the *relation* between phonological forms and feature bundles. Consider the pair of lexical items in table 6.

Table 6: Forms of (auxiliary) *have*

$\mathsf{has} :: \mathtt{perf}^+.\mathtt{k}^+.\mathtt{s}^-$	$\mathsf{have} :: \mathtt{perf}^+.\mathtt{v}^-$

Not only do both of these lexical items begin with a $\mathtt{perf}^+$ feature, but (we know) they are all forms of the auxiliary verb `have`. This generalization is, however, not expressed in the language of our theory (i.e. in terms of our lexicon). We would like to *factor out* the auxiliary from its endings. Abstractly, we need a decomposition rule as shown in figure 17. This rule allows a lexical item to be split into two; the feature bundle is broken into two as per the discussion above, but now the original phonological form of the lexical item (w) is factored into two pieces (u and v). These two new lexical items no longer give us the same structures we would have created with the original one, as now there are two heads (u and v) instead of just one

(w), and in addition all specifiers in α intervene between these two heads. In order to remedy this problem, we must allow u and v to combine to form w. We do this in two steps. First, when merging expressions headed by u and v respectively, we combine the phonological material from both heads, and put them together into just one of the two. Second, we record that these two heads are not pronounced separately as u and v, but are rather pronounced together as w. This is done in two ways. First, in the lexical entry for v (the selector), we record (by underlining the relevant feature) that it enters into a special relationship with the head of its selected argument ($\underline{\mathtt{x}^+}$). Second, we record that the lexical items u and v are jointly pronounced as w.[15] This latter is, in linguistic theory, the provenance of morphology. It is simply represented here as a finite list of statements (morphology can be thought of as a way of compressing this list, or alternatively a way of identifying and expressing rule-like generalizations). Using this decomposition rule, we factor out the lexical item have :: $\mathtt{perf}^+.\mathtt{y}^-$ from the two original have-forms in table 6, obtaining the lexical items in table 7.

Table 7: Factoring out *have*

have :: $\mathtt{perf}^+.\mathtt{y}^-$	v^{y} :: $\underline{\mathtt{y}^+}.\mathtt{v}^-$
	Pres :: $\underline{\mathtt{y}^+}.\mathtt{k}^+.\mathtt{s}^-$
have $\oplus$ Pres $= has$	have $\oplus$ $\mathsf{v}^{\mathsf{y}} = have$

The lexical item v^{y} :: $\underline{\mathtt{y}^+}.\mathtt{v}^-$ simply selects something of category y, and turns it into something of category v. Such a lexical item could be replaced by a partial ordering statement of the form $y \leq v$ asserting that a negative feature $\mathtt{y}^-$ can be used to satisfy a positive feature $\mathtt{v}^+$ (see Bernardi and Szabolcsi [2008]).

It is convenient to represent a morphological word forming edge between nodes differently from non-morphological word forming edges. I will use a dashed edge in all of our representational schemes to indicate this (it is redundant information, as it is reconstructable from the fact that an underlined feature was checked), as in figure 18.

[15] It would make sense as well to, upon combining u and v, to replace them with w. I prefer, when doing theory construction, to factor out logically distinct steps: syntax will then assemble complex heads, and these complex heads will be interpreted elsewhere. Of course, when actually ***using*** this theory to model performance, these logically distinct steps can and perhaps should be interleaved with one another.

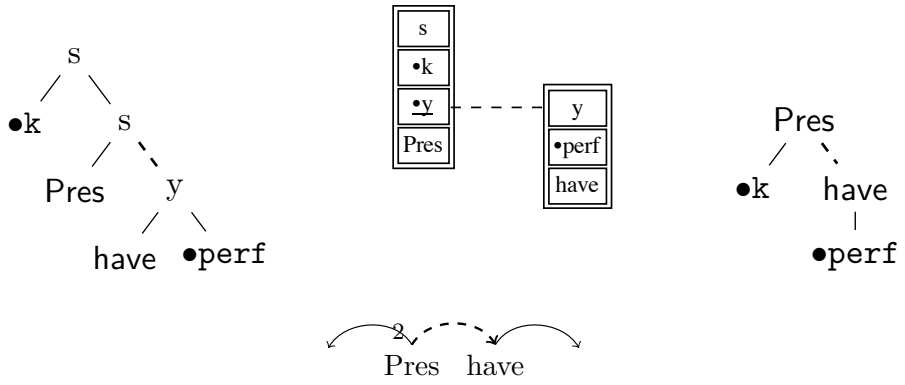


Figure 18: Visualizing morphological word forming dependencies

2.1.2 Decomposition and Generalization

The entire point about this sort of lexical decomposition can be summarized in the following way:

> lexical decomposition allows us to express regularities in the lexicon as new lexical items

If we measure the size of a grammatical description in terms of the number of features used (i.e. the sum of all features on all lexical items), then lexical decomposition can be used to reduce the size of our lexicon, by reifying repeated feature sequences as separate lexical items. Consider the set of lexical items in table 8.

Table 8: Some tensed lexical items

will :: $\mathtt{v}^+.\mathtt{k}^+.\mathtt{s}^-$	has :: $\mathtt{perf}^+.\mathtt{k}^+.\mathtt{s}^-$
is :: $\mathtt{prog}^+.\mathtt{k}^+.\mathtt{s}^-$	eats :: $\mathtt{d}^+.\mathtt{k}^+.\mathtt{s}^-$

Each of these four lexical items has three features, giving us a total lexical size of 12. However, all four lexical items end with the same length two sequence of features: $\mathtt{k}^+.\mathtt{s}^-$. This expresses that they check the case of a subject, and are a sentence; in even more naïve terms each lexical item demands that the subject moves to its specifier. We can decompose our lexical items so as to factor this common feature sequence out, assigning it to

a lexical item named AgrS. This gives rise to the lexicon in table 9. We need as well one morphological interpretation rule for each combination of lexical item with AgrS; as AgrS does not contribute to pronunciation (because we are not taking agreement into account). I abbreviate these rules as a single rule $\alpha \oplus \mathsf{AgrS} = \alpha$.

Table 9: Factoring out the feature sequence $\mathtt{k}^+.\mathtt{s}^-$

will :: $\mathtt{v}^+.\mathtt{t}^-$	has :: $\mathtt{perf}^+.\mathtt{t}^-$
is :: $\mathtt{prog}^+.\mathtt{t}^-$	eats :: $\mathtt{d}^+.\mathtt{t}^-$
AgrS :: $\underline{\mathtt{t}^+}.\mathtt{k}^+.\mathtt{s}^-$	$\alpha \oplus \mathsf{AgrS} = \alpha$

This lexicon has *five* lexical items, but only eleven features. We can see that decomposition has *reified* the repeated feature sequence as a new lexical item AgrS :: $\underline{\mathtt{t}^+}.\mathtt{k}^+.\mathtt{s}^-$, which expresses the generalization that subjects move to a specifier position at (or above) TP.

The desire to capture regularities in the lexicon is not limited to the transformational grammar tradition, but is common to nearly every linguistic endeavour. The tradition of head-driven phrase structure grammar [Pollard and Sag, 1994] makes extensive use of *type hierarchies* to express lexical regularities (see Flickinger [1987], Meurers [2001]), where recurring patterns in lexical entries are factored out into more abstract lexical entries. The left-over concrete lexical items are then connected to the abstract ones in an inheritance hierarchy, which specifies which information can be added to the concrete lexical items. Although the mechanisms involved are very different, one can see lexical decomposition as presented here as the Minimalist version of type hierarchies with inheritance being implemented via (complex head forming) merger. The graphical presentations of the lexicon as shown beginning in figure 20 can be understood as type hierarchies.

2.1.3 Complex heads

Once we move from whole word syntax to one which manipulates sub-word parts,[16] we must confront two questions.

1. how do the heads which constitute a single word get identified?
2. where does the word corresponding to multiple distinct heads get pronounced?

[16] In current parlance, this would be described as moving from a pre-syntactic morphological module to a post-syntactic one.

The answer to the first question we gave implicitly in the previous section: two heads are part of the same word just in case one selects for the other with an underlined feature. This is a version of a *spanning* approach to the syntax morphology interface Brody [2000], Williams [2003], Svenonius [2016]. There are many possible answers to the second question. Following Brody [2000], we say that a word is pronounced (relative to other words) as though it occupied the position of the highest of its heads (with respect to c-command) with a particular property (and in the lowest of its heads, if none have that property). This property is called *strength* in Brody's work, but is formally merely an *ad hoc* property of lexical items. To distinguish between strong and weak lexical items, we write strong lexical items with two colons separating their phonological and syntactic features, and weak ones with one (as in table 10).

Table 10: Strong (left) and weak (right) lexical items

Strong	**Weak**
$\mathsf{u} :: \alpha$	$\mathsf{v} : \beta$

Regular head movement is what happens when all lexical items are strong. More spell-out possibilities emerge when we extend the set of possible strengh values. Abels [2001] proposes adding a third value **undetermined**. An undetermined head behaves as does its morphological word forming daughter; if the daughter is strong, the undetermined head behaves as though it were strong. If the daughter is weak, the undetermined head behaves as though it were weak. Arregi and Pietraszko [to appear] propose a strength value we might call **sticky**, which holds the morphological word in place, unless it is pulled up by a higher sticky head. In the setting of Arregi and Pietraszko, all heads are either sticky or strong, and so a morphological word will be pronounced either in the highest sticky head, or in the highest head, if there are no sticky heads. Note that this is the same as the proposal of Brody (with **sticky** replacing **strong**, and **strong** replacing **weak**), except that the 'default' spellout position (i.e. the spellout position without any special **sticky**/**strong** nodes) is high, not low.

There is a deep similarity between the role of strength in determining the spell-out position of a complex head, and the role of (the formally unrelated but similarly named) strength in determining the spell-out position of a movement chain. This is because the two problems are fundamentally the same: there is an element (either the complex head, or a phrase), and multiple different positions in which it can be pronounced. *Crucially*, in both

cases, all of these positions are totally ordered, which permits the use of notions like 'highest.' It seems unparsimonious to implement the same solution to the same problem in two different ways (strength_1 on lexical items and strength_2 on features). There are two obvious ways to approach a unification of mechanisms, depending on which notion of strength we reduce to which other. Reducing lexical strength to featural strength means that we identify the strength of a lexical item (for the purposes of complex head spellout) with the strength of one of its features. It is most natural to identify this feature as the one checked in the service of forming the complex head. So the relevant feature of the lowest, root, position of the span is its first negative feature, and those of the higher positions in the span are their first positive features. Reducing featural strength to lexical strength means reconstructing featural strength on the basis of lexical strength. As a lexical item has but one value for its strength, but potentially many features, this entails that all features of a given lexical item would have the same strength value. While decomposition ensures that we can refactor lexical items so that each positive feature is associated with its own head, the same cannot be done for the negative features.[17] Thus, reducing featural strength to lexical strength would make useless the option of controlling chain spell-out with negative features.

2.1.4 Decomposing Auxiliaries

We thus want to express generalizations about our language in terms of our theory, and this we will do via decomposition. We want to compare lexical items to one another which share feature prefixes/suffixes and (ideally) similar phonologies. We should then decompose, and unify, decompose, and unify, until further decomposition does not achieve any succinctness gains.[18] However we will here simply note *en masse* that the `eat` verbs begin with the feature $\texttt{d}^+$, and decompose them. The result is shown in table 11

Note again that the original bare `eat` form has also been decomposed, leaving behind a 'dummy' lexical item (named v^V) which serves to simply change category. This is important, so that no new forms are derived: decomposition by itself does not change the language of the grammar. Were we not to do this, the newly decomposed `eat` lexeme would continue to have

[17] The obvious strategy of decomposing betwixt negative features runs afoul of the problem that it makes more features accessible than were originally.

[18] This is easier said than done! Marina Ermolaeva is exploring how MDL can be used to guide a search for the ideally decomposed lexicon. Preliminary results have been presented in Ermolaeva [2020].

Table 11: Lexical items before (left) and after (right) decomposition of *eat*

eat :: $\mathtt{d}^+.\mathtt{v}^-$	v^{V} :: $\underline{\mathtt{V}^+}.\mathtt{v}^-$
eats :: $\mathtt{d}^+.\mathtt{k}^+.\mathtt{s}^-$	Pres :: $\underline{\mathtt{V}^+}.\mathtt{k}^+.\mathtt{s}^-$
eaten :: $\mathtt{d}^+.\mathtt{perf}^-$	Perf :: $\underline{\mathtt{V}^+}.\mathtt{perf}^-$
eating :: $\mathtt{d}^+.\mathtt{prog}^-$	Prog :: $\underline{\mathtt{V}^+}.\mathtt{prog}^-$
	eat :: $\mathtt{d}^+.\mathtt{V}^-$
	eat $\oplus$ v^{V} $\mapsto$ *eat*
	eat $\oplus$ Pres $\mapsto$ *eats*
	eat $\oplus$ Perf $\mapsto$ *eaten*
	eat $\oplus$ Prog $\mapsto$ *eating*

category v, just as would the lexemes be and have. This would allow us to derive the ungrammatical "John will have been having been eating".

We next do the same with the forms of be.[19] This is shown in table 12.

Table 12: Lexical items before (right) and after (left) decomposition of *be*

be :: $\mathtt{prog}^+.\mathtt{v}^-$	v^{x} :: $\underline{\mathtt{x}^+}.\mathtt{v}^-$
is :: $\mathtt{prog}^+.\mathtt{k}^+.\mathtt{s}^-$	Pres :: $\underline{\mathtt{x}^+}.\mathtt{k}^+.\mathtt{s}^-$
been :: $\mathtt{prog}^+.\mathtt{perf}^-$	Perf :: $\underline{\mathtt{x}^+}.\mathtt{perf}^-$
	be :: $\mathtt{prog}^+.\mathtt{x}^-$
	be $\oplus$ v^{x} $\mapsto$ *be*
	be $\oplus$ Pres $\mapsto$ *is*
	be $\oplus$ Perf $\mapsto$ *been*

Comparing the results of decomposing eat on the one hand and be on the other, we see that we have three pairs of abstract lexical items (those named v^{α}, Pres, and Perf) which have identical feature bundles but for the names of their first features: V versus x. There are three basic possibilities for systematically relating any of these pairs:

1. identify them, thereby unifying V and x (here I have renamed x as V)

[19] There is a deep issue here, regarding how we are to know that is is a form of be. There has been computational work on identifying morphological paradigms [Lee, 2014], which might very well be of use here.

before	*after*
v^{x} :: $\underline{\mathtt{x}^{+}}.\mathtt{v}^{-}$	
v^{V} :: $\underline{\mathtt{V}^{+}}.\mathtt{v}^{-}$	v^{V} :: $\underline{\mathtt{V}^{+}}.\mathtt{v}^{-}$
Pres :: $\underline{\mathtt{x}^{+}}.\mathtt{k}^{+}.\mathtt{s}^{-}$	
Pres :: $\underline{\mathtt{V}^{+}}.\mathtt{k}^{+}.\mathtt{s}^{-}$	Pres :: $\underline{\mathtt{V}^{+}}.\mathtt{k}^{+}.\mathtt{s}^{-}$
Perf :: $\underline{\mathtt{x}^{+}}.\mathtt{perf}^{-}$	
Perf :: $\underline{\mathtt{V}^{+}}.\mathtt{perf}^{-}$	Perf :: $\underline{\mathtt{V}^{+}}.\mathtt{perf}^{-}$

2. decompose the first into something else plus the second, thereby asserting that $V \leq x$

before	*after*
v^{x} :: $\underline{\mathtt{x}^{+}}.\mathtt{v}^{-}$	v^{x} :: $\underline{\mathtt{x}^{+}}.\mathtt{v}^{-}$
v^{V} :: $\underline{\mathtt{V}^{+}}.\mathtt{v}^{-}$	
Pres :: $\underline{\mathtt{x}^{+}}.\mathtt{k}^{+}.\mathtt{s}^{-}$	Pres :: $\underline{\mathtt{x}^{+}}.\mathtt{k}^{+}.\mathtt{s}^{-}$
Pres :: $\underline{\mathtt{V}^{+}}.\mathtt{k}^{+}.\mathtt{s}^{-}$	
Perf :: $\underline{\mathtt{x}^{+}}.\mathtt{perf}^{-}$	Perf :: $\underline{\mathtt{x}^{+}}.\mathtt{perf}^{-}$
Perf :: $\underline{\mathtt{V}^{+}}.\mathtt{perf}^{-}$	
	x^{V} :: $\underline{\mathtt{V}^{+}}.\mathtt{x}^{-}$

3. decompose the second into something else plus the first, thereby asserting that $x \leq V$

before	*after*
v^{x} :: $\underline{\mathtt{x}^{+}}.\mathtt{v}^{-}$	
v^{V} :: $\underline{\mathtt{V}^{+}}.\mathtt{v}^{-}$	v^{V} :: $\underline{\mathtt{V}^{+}}.\mathtt{v}^{-}$
Pres :: $\underline{\mathtt{x}^{+}}.\mathtt{k}^{+}.\mathtt{s}^{-}$	
Pres :: $\underline{\mathtt{V}^{+}}.\mathtt{k}^{+}.\mathtt{s}^{-}$	Pres :: $\underline{\mathtt{V}^{+}}.\mathtt{k}^{+}.\mathtt{s}^{-}$
Perf :: $\underline{\mathtt{x}^{+}}.\mathtt{perf}^{-}$	
Perf :: $\underline{\mathtt{V}^{+}}.\mathtt{perf}^{-}$	Perf :: $\underline{\mathtt{V}^{+}}.\mathtt{perf}^{-}$
	V^{x} :: $\underline{\mathtt{x}^{+}}.\mathtt{V}^{-}$

Pursuing options 1 or 3 would collapse necessary syntactic distinctions, leading the grammar to generate sentences of the form: {John will be (being)* eating}. The correct option is 2. This can be determined in a less intuitive manner by identifying cycles in selection (or the lack thereof) in the lexicon: a V can be turned into a prog (via ing), which can be turned into an x (via be), but an x cannot become a V.[20] Adding this information (as an empty lexical item) to our lexicon gives us the lexicon in table 13.

[20]This is unfortunately quite a bit more complicated (see Ermolaeva [2020]). An x could in principle become a V, if it were embedded as a clausal complement, or as a relative clause modifier of a DP complement, of a higher verb.

Table 13: Lexical items after decomposing be and eat, and asserting that $V \leq x$

x^{V} :: $\underline{\mathtt{V}^{+}}.\mathtt{x}^{-}$	Pres :: $\underline{\mathtt{x}^{+}}.\mathtt{k}^{+}.\mathtt{s}^{-}$
v^{x} :: $\underline{\mathtt{x}^{+}}.\mathtt{v}^{-}$	Perf :: $\underline{\mathtt{x}^{+}}.\mathtt{perf}^{-}$
	Prog :: $\underline{\mathtt{V}^{+}}.\mathtt{prog}^{-}$
eat :: $\mathtt{d}^{+}.\mathtt{V}^{-}$	be :: $\mathtt{prog}^{+}.\mathtt{x}^{-}$
$\mathsf{eat} \oplus \mathsf{x}^{\mathsf{V}} \oplus \mathsf{Pres} = \mathit{eats}$	$\mathsf{be} \oplus \mathsf{Pres} = \mathit{is}$
$\mathsf{eat} \oplus \mathsf{x}^{\mathsf{V}} \oplus \mathsf{Perf} = \mathit{eaten}$	$\mathsf{be} \oplus \mathsf{Perf} = \mathit{been}$
$\mathsf{eat} \oplus \mathsf{x}^{\mathsf{V}} \oplus \mathsf{v}^{\mathsf{x}} = \mathit{eat}$	$\mathsf{be} \oplus \mathsf{v}^{\mathsf{x}} = \mathit{be}$
$\mathsf{eat} \oplus \mathsf{Prog} = \mathit{eating}$	

A similar issue arises upon decomposing have, which we did back in table 7. We see again that we have two pairs of very similar looking lexical items (v^{x} :: $\underline{\mathtt{x}^{+}}.\mathtt{v}^{-}$ and v^{y} :: $\underline{\mathtt{y}^{+}}.\mathtt{v}^{-}$ on the one hand, and Pres :: $\underline{\mathtt{x}^{+}}.\mathtt{k}^{+}.\mathtt{s}^{-}$ and Pres :: $\underline{\mathtt{y}^{+}}.\mathtt{k}^{+}.\mathtt{s}^{-}$ on the other). Wanting to relate these lexical items, we have the three familiar choices: identify x and y, set $x \leq y$, or set $y \leq x$. Allowing $\mathtt{y}^{-}$ to satisfy a $\mathtt{x}^{+}$ feature would allow us to derive ungrammatical sentences like "John will have (had)* eaten." Thus, we cannot identify x and y, nor can we set $y \leq x$.

before	*after*
v^{y} :: $\underline{\mathtt{y}^{+}}.\mathtt{v}^{-}$	v^{y} :: $\underline{\mathtt{y}^{+}}.\mathtt{v}^{-}$
v^{x} :: $\underline{\mathtt{x}^{+}}.\mathtt{v}^{-}$	
Pres :: $\underline{\mathtt{y}^{+}}.\mathtt{k}^{+}.\mathtt{s}^{-}$	Pres :: $\underline{\mathtt{y}^{+}}.\mathtt{k}^{+}.\mathtt{s}^{-}$
Pres :: $\underline{\mathtt{x}^{+}}.\mathtt{k}^{+}.\mathtt{s}^{-}$	
	y^{x} :: $\underline{\mathtt{x}^{+}}.\mathtt{y}^{-}$

Table 14: Lexical items after decomposing be, eat, and have, and asserting that $V \leq x$ and $x \leq y$

v^{y} :: $\underline{\mathtt{y}^{+}}.\mathtt{v}^{-}$	y^{x} :: $\underline{\mathtt{x}^{+}}.\mathtt{y}^{-}$	x^{V} :: $\underline{\mathtt{V}^{+}}.\mathtt{x}^{-}$	eat :: $\mathtt{d}^{+}.\mathtt{V}^{-}$
will :: $\mathtt{v}^{+}.\mathtt{k}^{+}.\mathtt{s}^{-}$	have :: $\mathtt{perf}^{+}.\mathtt{y}^{-}$	be :: $\mathtt{prog}^{+}.\mathtt{x}^{-}$	John :: $\mathtt{d}^{-}.\mathtt{k}^{-}$
Pres :: $\underline{\mathtt{y}^{+}}.\mathtt{k}^{+}.\mathtt{s}^{-}$	Perf :: $\underline{\mathtt{x}^{+}}.\mathtt{perf}^{-}$	Prog :: $\underline{\mathtt{V}^{+}}.\mathtt{prog}^{-}$	

This lexicon has 24 features in it (18, if we discount the **isa** lexical items), whereas the initial lexicon (prior to decomposition) contained 26 features.

Table 15: Morphological equations for table 14

$\mathsf{have} \oplus \mathsf{Pres} = has$	$\mathsf{be} \oplus \mathsf{y}^{\mathsf{x}} \oplus \mathsf{Pres} = is$	$\mathsf{eat} \oplus \mathsf{x}^{\mathsf{V}} \oplus \mathsf{Pres} = eats$
$\mathsf{have} \oplus \mathsf{v}^{\mathsf{y}} = have$	$\mathsf{be} \oplus \mathsf{y}^{\mathsf{x}} \oplus \mathsf{v}^{\mathsf{y}} = be$	$\mathsf{eat} \oplus \mathsf{x}^{\mathsf{V}} \oplus \mathsf{y}^{\mathsf{x}} \oplus \mathsf{v}^{\mathsf{y}} = eat$
	$\mathsf{be} \oplus \mathsf{Perf} = been$	$\mathsf{eat} \oplus \mathsf{x}^{\mathsf{V}} \oplus \mathsf{Perf} = eaten$
		$\mathsf{eat} \oplus \mathsf{Prog} = eating$

We have thus achieved a (small) compression. However, the important difference between these two lexica lies in their behaviour as more words are added to them; novel intransitive verbs contribute just two features to our final lexicon, but 9 features (distributed over four lexical items) to our initial one.

2.2 More on English auxiliaries

Our analysis of sentences 1 — 8 culminated in the set of lexical items presented in table 14, together with the morphological equations in table 15. Sets of lexical items, although the objects in terms of which analyses are presented, are not the most perspicuous objects for communicating these analyses. Linguists typically present structures in order to communicate their analyses, although these structures only ever exemplify a particular way of combining lexical items. In figure 20 I present a lexical graph, which describes all the combinatory possibilities of the lexicon in table 14. The original, word based analysis of the same sentences is given in figure 19. Comparing the two figures, one sees that there are fewer nodes in the graph for the original analysis (four versus seven) – decomposition has created three additional abstract categories y, x, and V. One also sees that each morphological form of the word `eat` corresponds to an edge in the original analysis, whereas there is a single edge for `eat` in the decomposed analysis (along with a similar pattern for the auxiliaries `have` and `be`). This is the basis of the claim that the original analysis requires four lexical entries for a single intransitive verb, whereas the decomposed one just one.

Reading this flowchart from left to right, we can see that it expresses the generalizations that:

1. (present) tense and modals (`will`) are first, and in complementary distribution

2. Next, perfective `have` may appear, if it does, then whatever is next will be in the perfective form

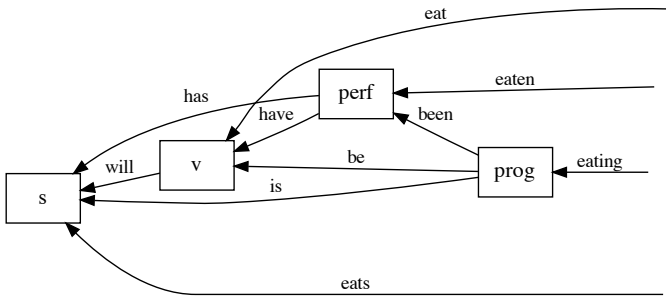


Figure 19: Depicting relations between lexical items for the English auxiliary system, before decomposition

3. Next, progressive `be` may appear, if it does then whatever is next will be in the progressive form

The flowchart notation will be used frequently in this paper, and so I will explain how to interpret it in some detail. The flowchart is a graph, where the nodes (shown as boxes) represent grammatical categories, and the edges (directed arrows) represent lexical items. Formally, an edge w exists from node u to node v just in case the first feature of the lexical item represented by w is $\mathtt{u}^+$, and the first negative feature of this lexical item is $\mathtt{v}^-$.[21] There is an edge will from node v to node s in the above graph because the lexical item will :: $\mathtt{v}^+.\mathtt{k}^+.\mathtt{s}^-$ has as its first feature $\mathtt{v}^+$ and its first negative feature is $\mathtt{s}^-$.

The analysis represented in figure 20 was arrived at in a quasi-mechanical way, by decomposing lexical items in order to try to eliminate redundancies in the lexicon. However, the analysis is only for a *fragment* of English (even of the English auxiliary system), and we will try to extend it in a similarly mechanical way. The lexical items which form complex heads (those whose

[21] This sort of graph just encodes head-complement relationships, and thus represents just a portion of the information contained in a lexicon. This suffices for our present purposes, as we are here primarily investigating the clausal spine. In general, a lexical graph is a directed hypergraph, where nodes correspond to feature types, and where each lexical item is represented as a hyperedge, whose sequence of source nodes are the positive features in its bundle, and whose sequence of target nodes are the negative features in its bundle.

first feature is of the form $\underline{\mathtt{x}^+}$ for some x) are represented by means of dotted edges in the figure.

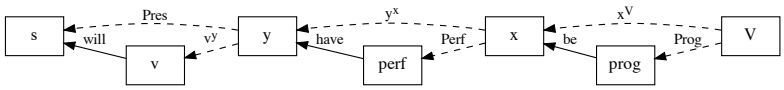


Figure 20: Depicting relations between lexical items for the English auxiliary system, after decomposition

Before we continue, I will modify our lexicon so as to collapse the distinction between the categories v and y by renaming them both to y. This will make the lexical item represented as an arrow from y to v a silent loop, which we will want to eliminate. This lexical item was introduced when decomposing the forms of the word `have`, but has outlived its usefulness. I will then, now that v is unused, change the category named V to v.

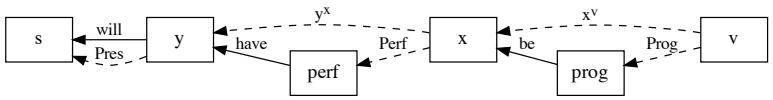


Figure 21: Streamlining our analysis

2.2.1 Tense and modals

We have in our fragment only a single tense, the present, and a single modal, `will`. We would like to extend our fragment so as to deal with more sentences, such as the below.

9. John ate
10. John would eat
11. John would have eaten
12. John must have been eating
13. John must be eating

$\mathsf{Past} :: \underline{\mathtt{y}^+}.\mathtt{k}^+.\mathtt{s}^-$ $\mathsf{would} :: \mathtt{y}^+.\mathtt{k}^+.\mathtt{s}^-$ $\mathsf{must} :: \mathtt{y}^+.\mathtt{k}^+.\mathtt{s}^-$

$\mathsf{eat} \oplus \mathsf{x}^{\mathsf{v}} \oplus \mathsf{y}^{\mathsf{x}} \oplus \mathsf{Past} = ate$

Table 16: New lexical items and equations

14. John must eat

As before, we can analyze these sentences as being composed out of whole words (given the obvious dependency structures), and decompose them to identify regularites. To save time, I will simply report the results of this process in table 16.

These lexical items all fall into familiar position classes; namely Past is parallel to Pres, and would and must are parallel to will in the sense that they have the same feature bundles (and thus the same distribution). We can see this by looking at the flowchart for our lexicon.

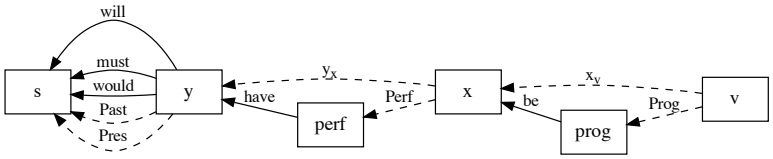


Figure 22: Modals vs Tense

There is one major locus of redundancy in the grammar: all of these tense and modal elements share a feature sequence: $\mathtt{k}^+.\mathtt{s}^-$. Accordingly, we extract a shared functional head $\mathsf{AgrS} :: \underline{\mathtt{t}^+}.\mathtt{k}^+.\mathtt{s}^-$ from our tense/modal elements, as shown excerpted below in figure 17. In this table is an unfamiliar

Table 17: (Part of) the lexicon after severing the subject position from inflection

$\mathsf{AgrS} :: \underline{\mathtt{t}^+}.\mathtt{k}^+.\mathtt{s}^-$ $\mathsf{will} :: \mathtt{y}^+.\mathtt{t}^-$
$\mathsf{Pres} :: \underline{\mathtt{y}^+}.\mathtt{t}^-$

$\mathsf{will} \oplus \mathsf{AgrS} = will$ $\mathsf{Pres} \mapsto \mathsf{Pres} \oplus \mathsf{AgrS}$

notation: $\mathsf{Pres} \mapsto \mathsf{Pres} \oplus \mathsf{AgrS}$. This is to be understood as saying that in all

of the morphological equations in our grammar, we replace the present tense Pres (the left hand side) with the present tense Pres plus the head hosting the subject position. Of course, we need other equations for the other modals, and another substitution for the past tense Past, which are not shown here.

This gives rise to a lexicon displayed in the flowchart below in figure 23. In this lexicon, each one of the eight tense and modal elements have one fewer features, and in exchange we have added a new lexical item with three features, making for a net decrease of five features.

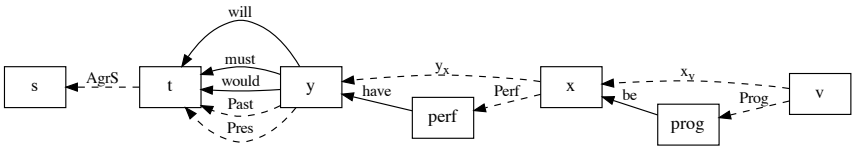


Figure 23: Severing the subject position from inflection

There are no longer any featural redundancies in our lexicon, but we can reduce the number of lexical entries if we view pairs like will and would as being inflected forms of a single lexeme.[22] We can decompose the tenses out from those lexical items, as well as from our current tense lexical items, and obtain the lexical items below.

Table 18: Some lexical items after decomposing modals and tense

$\mathsf{Past} :: \underline{\mathtt{m}^+}.\mathtt{t}^-$	$\mathsf{m}^{\mathsf{y}} :: \underline{\mathtt{y}^+}.\mathtt{m}^-$
	$\mathsf{will} : \underline{\mathtt{y}^+}.\mathtt{m}^-$
$\mathsf{Past} \mapsto \mathsf{m}^{\mathsf{y}} \oplus \mathsf{Past}$	$\mathsf{will} \oplus \mathsf{Past} = \mathit{would}$

The lexicon we arrive at is easier to make sense of as a flowchart, which is displayed in figure 24.

[22] Some modals, like must, doesn't appear to have a 'tense' distinction. Instead of listing this fact in the morphology (must $\oplus$ Pres $= \mathit{must}$ and must $\oplus$ Past $= \mathit{must}$) I have opted for making must 'skip over' the tense head.

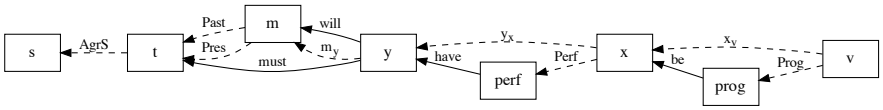


Figure 24: Decomposing modals and tense

2.2.2 Negation and `do`

The English affixal negation `n't` has been argued to be an inflectional affix rather than a cliticized form of the negative word `not` [Zwicky and Pullum, 1983]. I adopt this analysis here. We begin again with a set of sentences we would like to extend our grammar developed thus far to account for.

15. John won't have been eating
16. John hasn't eaten
17. John isn't eating
18. John mustn't eat

Beginning once again with a whole word analysis, we can successively decompose the novel forms (won't, hasn't, isn't, and mustn't) until we obtain the negative head nt, and the new category *pol*.

Table 19: The polarity lexical items

$\mathsf{nt} :: \underline{\mathtt{t}^+}.\mathtt{pol}^-$	$\mathsf{pol}^{\mathsf{t}} :: \underline{\mathtt{t}^+}.\mathtt{pol}^-$
$\mathsf{will} \oplus \mathsf{Pres} \oplus \mathsf{nt} = won't$	$\mathsf{have} \oplus \mathsf{m}^{\mathsf{y}} \oplus \mathsf{Pres} \oplus \mathsf{nt} = hasn't$

The flowchart for the resulting lexicon is displayed in figure 25. I have truncated everything to the right of the category y, so as to focus on the area of interest (the 'IP' domain).

Sentence 15 shows us that nt must be introduced above have and be. Sentence 18 shows us that nt must not interact with the category m. The options then for the placement of nt are restricted to the categories y, t and s. We have chosen the second option, that of splitting t into two categories, t and *pol*, with nt connecting the two. Choosing the first option, of splitting

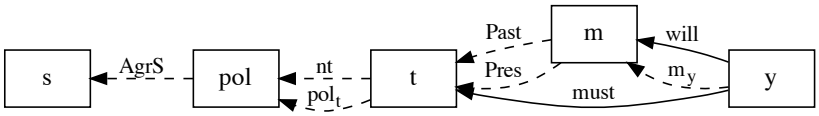


Figure 25: Introducing negation

what is currently y in to y and *pol* would be formally possible, but as a consequence we would need to abandon the generalization that the 'head' of the morphological word is introduced syntactically low; then `n't` would be the first head making up the word `won't`, and thus will would need to be reanalyzed as an affix. This is well within the realm of analytical possibilities, but departs wildly from linguistic dogma. The third option, of splitting the category s into s and *pol* would require us to make the negation lexical item *weak* - it could not support spellout of the morphological word it is part of. This is because the subject position is introduced by the empty head leading in to the category s, and subjects are pronounced to the left of negative word forms in our data. Thus far, we have not had to worry about the strength of our lexical items; we have simply assumed them all to be strong.[23] Our choice of the second option (above tense, but below the subject position) is in line with much syntactic work, which places negation high in the clause between the surface subject position (in AgrSP) and tense [Pollock, 1989].

This analysis predicts the syntactic existence of forms like `eatsn't` and `aten't`, which of course do not exist. These unattested forms are predicted by our analysis because negation combines with tense, irrespective of what may have come before. This is a standard problem, and a standard move in transformational grammar is to impose a filter that blocks those structures in which the verb has combined with affixal negation. One way to do this is to prohibit the verb from raising through negation, i.e. by banning formation of a complex head containing both negation and a main verb. Auxiliaries and modals are exempt from such a filter. This will be expanded in section 2.2.3.

Instead of using affixal negation on main verbs, English makes use of the periphrastic alternatives below.

[23]This is far from a knock down argument. In fact, most people seem to want lexical verbs to be pronounced rather low in the structure, which would require the following strength settings on our lexical items (speaking in the language of figure 25): the empty lexical items between v and x, between x and y, and between y and m must be weak, as too must the tense morphemes. Then the polarity morphemes that we have introduced should also be weak (whether they are placed at t or at s).

19. John doesn't eat

20. John didn't eat

Again we begin with a whole-word analysis (the details of which are again suppressed), and end up with the additional lexical item in table 20. Lexical

Table 20: lexical *do*

$$\texttt{do} :: \texttt{v}^{+}.\texttt{m}^{-}$$

do has been assigned the feature bundle $\texttt{v}^{+}.\texttt{m}^{-}$ because it

1. combines only with an uninflected lexical verb ($\texttt{v}^{+}$)

2. is compatible with tense ($\texttt{m}^{-}$) but not with modals, or auxiliaries

The flowchart for our expanded lexicon is given in figure 26. I have here truncated everything to the left of *t*, thus focussing attention on the verbal domain.

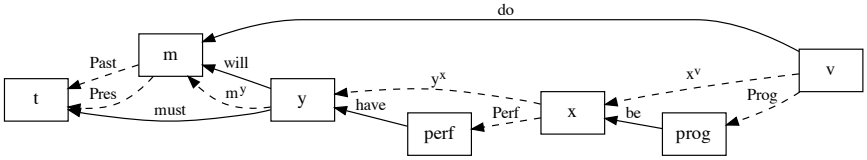


Figure 26: Introducing do

In order to better describe *do*-support in the next section, it will be useful to expand our grammar fragment to cover more constructions in which do occurs, such as verum focus (sentence 21) sentences and subject-auxiliary inversion contexts (sentence 22) without auxiliaries or modals.[24]

21. John DIDN'T eat.

22. Did John eat?

[24] *Do* also appears in VP-ellipsis contexts and negative imperatives. We will not discuss ellipsis in this chapter (see Kobele [2015] for an attempt at an elegant yet computationally tractable take on mainstream generative approaches thereto). I assume the appearance of *do* in negative imperatives will yield to whatever account suffices for its appearance in negative sentences more generally.

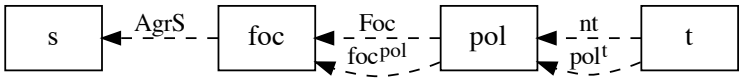


Figure 27: Adding verum focus

To deal with verum focus (as in 21), we begin again with whole word analyses of the sentences in question, obtaining focussed auxiliaries like MUST :: $\mathtt{y}^+.\mathtt{k}^+.\mathtt{s}^-$ and DIDN'T :: $\mathtt{v}^+.\mathtt{k}^+.\mathtt{s}^-$. We 'know' (given our intuitions and analysis to date) that inside MUST we should find must, $\mathsf{pol}^{\mathsf{t}}$, and AgrS, and that inside DIDN'T we should find do, Past, nt, and AgrS, alongside a new Foc head. Because in our analysis must excludes tense, this Foc head must itself be above tense, meaning that in order to unify the Foc heads of MUST and DIDN'T we must extract them from these words prior to decomposing tense from DIDN'T. Nothing forces an ordering of decomposition steps between Foc and AgrS or the polarity heads, however. If we decompose Foc *first* from MUST and DIDN'T, then it will be in a position *higher* than the head hosting the subject position (AgrS), and thus *must* be weak (on pain of having the inflected unit precede the subject). Decomposing Foc *after* decomposing AgrS means that the strengh of Foc is irrelevant. I will simply decide to decompose Foc immediately after AgrS, resulting in the lexical item Foc :: $\underline{\mathtt{pol}^+}.\mathtt{foc}^-$. Alongside the semantically contentful Foc head, we also need a type changing head $\mathsf{foc}^{\mathsf{pol}}$ implementing the ordering statement $\mathsf{pol} \leq \mathsf{foc}$, allowing for sentences without verum focus. This is shown in figure 27.

The reasoning behind subject auxiliary inversion is quite parallel to the above. Except that in this case we *want* the inflected item to precede the subject, and so must decompose a SAI head (which I will call SAI) above AgrS. Renaming categories, we obtain the lexical item SAI :: $\underline{\mathtt{s}^+}.\mathtt{c}^-$ (and so we have renamed the category s to c, and the new category created by decomposing SAI is now called $\mathtt{s}^-$). Again, we must also introduce the type changing changing head c^{s} implementing the ordering statement $\mathsf{s} \leq \mathsf{c}$, allowing for sentences without subject auxiliary inversion. This head however *must* be weak. This is shown in figure 28.

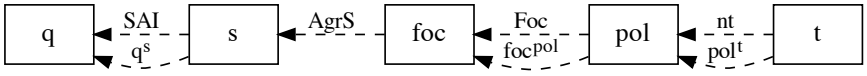


Figure 28: Adding subject auxiliary inversion

Our final analysis for the English auxiliary system can be viewed as the

flowchart in figure 29. This flowchart corresponds to the lexicon in table 21,

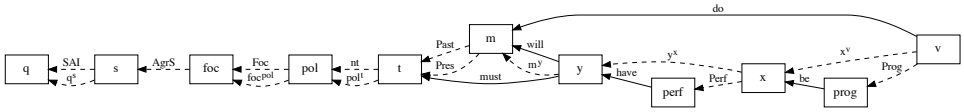


Figure 29: The English auxiliary system

which assigns strength to lexical items in such a manner as to realize verbs low in the structure, or to the variant of this lexicon where all heads are uniformly strong. The lexical items in the table have been made weak by default. This makes every lexeme be realized in its base position, unless of course the subject-auxiliary inversion head is present. There is no strength setting on heads in this analysis that will make auxiliaries be pronounced uniformly in T, and lexical verbs in V. If we adapt the proposal of Arregi and Pietraszko [to appear], and use in addition to **weak** the strengh value **sticky**, then if all lexical items are strong, except for lexical verbs, which are sticky, and the non-SAI head c^{s}, which is weak, then lexical verbs will always be pronounced in their base positions, whereas auxiliaries, modals and *do* will be pronounced in their highest positions.

Table 21: Lexical items for the English auxiliary system

SAI : : $\underline{\mathtt{s}^+}.\mathtt{c}^-$	nt : $\underline{\mathtt{t}^+}.\mathtt{pol}^-$	do : $\mathtt{v}^+.\mathtt{m}^-$	have : $\mathtt{perf}^+.\mathtt{y}^-$
c^{s} : $\underline{\mathtt{s}^+}.\mathtt{c}^-$	$\mathsf{pol}^{\mathsf{t}}$: $\underline{\mathtt{t}^+}.\mathtt{pol}^-$	will : $\mathtt{y}^+.\mathtt{m}^-$	Perf : $\underline{\mathtt{x}^+}.\mathtt{perf}^-$
AgrS : $\underline{\mathtt{foc}^+}.\mathtt{k}^+.\mathtt{s}^-$	Past : $\underline{\mathtt{m}^+}.\mathtt{t}^-$	m^{y} : $\underline{\mathtt{y}^+}.\mathtt{m}^-$	be : $\mathtt{prog}^+.\mathtt{x}^-$
Foc : $\underline{\mathtt{pol}^+}.\mathtt{foc}^-$	Pres : $\underline{\mathtt{m}^+}.\mathtt{t}^-$	y^{x} : $\underline{\mathtt{x}^+}.\mathtt{y}^-$	Prog : $\underline{\mathtt{v}^+}.\mathtt{prog}^-$
$\mathsf{foc}^{\mathsf{pol}}$: $\underline{\mathtt{pol}^+}.\mathtt{foc}^-$	must : $\mathtt{y}^+.\mathtt{t}^-$	x^{v} : $\underline{\mathtt{v}^+}.\mathtt{x}^-$	eat : $\mathtt{d}^+.\mathtt{v}^-$

2.2.3 *Do*-support

The decompositional methodology relies not just on decomposing lexical items, but also on unifying lexical items that seem like they are copies of each other. For example, when decomposing `eats` into the root eat and a tense head Pres, and `is` into the root be and a tense head Pres, I made a decision to treat both tense heads as the same. This decision was not discussed much, but it was very consequential. Although decomposition itself does not change the language generated by a lexicon, unifying categories (or lexical items) *can*. In particular, `is` has a different distribution in the language than `eats` - it may be inverted with respect to the subject (as in questions) and it

may appear before negation. By unifying the tensed Pres of `is` with the tensed Pres of `eats`, we are giving both forms the same derivational future. Similarly, by treating the Pres of `does` as the same as the Pres of `eats` and `is`, we give them again the same derivational future. Once we have made this step, there is no syntactic route back, and so we need to appeal to some other mechanism to rein in our self-inflicted overgeneration.

Table 22: A characterization of *do*-support

if do **then** nt **or** Foc **or** SAI

if $lexV$ **then not** (nt **or** Foc **or** SAI)

Carefully examining our overgeneration in light of our theory, we observe a complementarity, summarized in table 22. The unusual descriptions in the table are intended to be predicates of paths through the flowchart in figure 29: for any lexical item ℓ, the atomic predicate ℓ holds of a path just in case that path contains that lexical item (i.e. just in case that path goes along an edge representing that lexical item). The variable $lexV$ is a meta-variable over lexical verbs (which in our current lexicon are just the solitary eat). The paths that overgenerate are those which do not satisfy either predicate. The complimentarity of distribution between lexical verbs and do is explicit in these predicates, with the consequents being negations of one another. These predicates state on the one hand that if do is used, then at least one of nt, Foc, or SAI must also be used. Therefore, do must be blocked if none of those heads appear. This can be implemented by the morphology refusing to interpret the following sequences of heads: $\mathsf{do} \oplus Tns \oplus \mathsf{pol}^{\mathsf{t}} \oplus \mathsf{foc}^{\mathsf{pol}} \oplus \mathsf{AgrS} \oplus \mathsf{c}^{\mathsf{s}}$ (here, Tns is a metavariable over the tense heads Pres and Past). On the other, if a lexical verb appears, then the opposite must obtain: neither nt, nor Foc, nor SAI may be used. Therefore, a lexical verb must be blocked if any of these heads appear. Formulating this as a filter (a ban on illicit paths) requires three independent statements, banning lexical verbs appearing with negation: $\mathsf{finiteVerb} \oplus \mathsf{nt}$, or with verum focus: $\mathsf{finiteVerb} \oplus \mathsf{pol}^{\mathsf{t}} \oplus \mathsf{Foc}$, or with subject auxiliary inversion: $\mathsf{finiteVerb} \oplus \mathsf{pol}^{\mathsf{t}} \oplus \mathsf{foc}^{\mathsf{pol}} \oplus \mathsf{AgrS} \oplus \mathsf{SAI}$. Here, finiteVerb is an abbreviation for $lexV \oplus \mathsf{x}^{\mathsf{v}} \oplus \mathsf{y}^{\mathsf{x}} \oplus \mathsf{m}^{\mathsf{y}} \oplus Tns$.

Of course, the complementarity of distribution between do and lexical verbs seems like an accident if our account is in terms of morphological (or derivational) filters. Rephrasing this objection in terms of the present paper: having two independent statements when one would do results in a grammatical description which is redundant (i.e. larger than necessary). This

however is not one that our current redundancy elimination techniques (of decomposition and unification) can deal with, and so we must rely on brute ingenuity. A popular strategy is to treat (auxiliary) do not as a lexical item, but as the morphological realization of an otherwise illegitimate complex head. Then, if illegitimate complex heads only arise when lexical verbs appear with negation, verum focus, or SAI, then we have an elegant explanation for the complementarity of distribution between do and lexical verbs. This idea can be implemented straightforwardly, but we need to be careful. We cannot simply pronounce an illegitimate complex head (containing a lexical verb!) as a form of do, because the lexical verb *does* appear after all (though uninflected) in sentences with *do*-support. Instead we must change our perspective on complex head formation - having your category feature checked by an underlined feature is still *necessary* to being a complex word, but can no longer be *sufficient*. We can stipulate that the heads nt, Foc, and SAI, if they enter in to a relationship with a sequence of heads containing a lexical verb, forcibly detach the lexical verb from the complex word sequence. This is more related to our filtering strategy than it might at first appear. We in effect continue to have the (now necessarily) morphological filters on finite verbs described above, but now instead of simply refusing to interpret them, we have introduced a *repair* operation which breaks up an uninterpretable complex head like $lexV \oplus \mathsf{x}^{\mathsf{v}} \oplus \mathsf{y}^{\mathsf{x}} \oplus \mathsf{m}^{\mathsf{y}} \oplus Tns \oplus \mathsf{nt}$ into two interpretable ones: $lexV$ and $\mathsf{do} \oplus Tns \oplus \mathsf{nt}$. Note that I have not only broken off the $lexV$ from this complex head, but I have also replaced the subsequence $\mathsf{x}^{\mathsf{v}} \oplus \mathsf{y}^{\mathsf{x}} \oplus \mathsf{m}^{\mathsf{y}}$ with do. By placing the repair operation at the interface, the morphology receives only inputs that it can interpret normally.

It is possible to think of this as an instance of *periphrastic exponence* [Blevins], where the syntax passes an input (in this case an uninterpretable head like $\mathsf{eat} \oplus \mathsf{x}^{\mathsf{v}} \oplus \mathsf{y}^{\mathsf{x}} \oplus \mathsf{m}^{\mathsf{y}} \oplus \mathsf{Pres} \oplus \mathsf{nt}$) to the interface, and receives multiple words (doesn't and eat) to linearize.

2.3 Grammatical function changing operations

Previously we attempted to provide an analysis of the English auxiliary system on methodological first principles. Instead of appealing to theoretical desiderata, such as having a fixed set of functional projections, in a certain order, etc, we began with whole words, and tried systematically to reduce redundancy in our analysis by means of our operation of lexical decomposition. Lexical decomposition is an operation on lexical items which splits one lexical item into two, by dividing its feature bundle into two parts, and giving one part to one, and the other part to the other. We saw that lexical

decomposition can sometimes end up *decreasing* the size of our lexicon, both in terms of the raw number of lexical items used (because parts of lexical items can be reused) but also in terms of the total number of features used.

In this section we will continue with this methodology, looking at more constructions, specifically, adding passivization (and consequently transitive verbs) and raising (both object and subject) to the mix.

2.3.1 Raising to subject

Our analysis thus far does not generate sentences like the below.

24. John will seem to laugh.

25. Mary seemed to have been crying.

I assume the dependencies between whole words shown in figure 30.

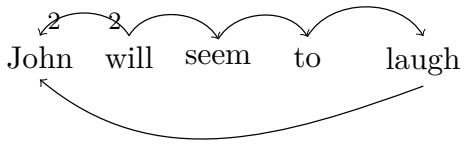


Figure 30: Dependencies for sentence 'John will seem to laugh

In other words, there is a (selectional) dependency between the embedded verb and the subject, but none between the (uninflected) verb `seem` and the subject.

In lieu of explicitly decomposing and unifying, I will discuss the process at a high level. The word `to` in our corpus has the same range of selectional possibilities as does `must`; it may combine with bare verbs, with `have`, or with `be`. It does not combine with `will` or `must`. Thus, it must select something of category y. In turn, it itself is not selected by will (and so cannot result in something of category y), nor does it inflect (and so cannot result in something of category m), nor can it be affixally negated (and so cannot result in something of category t), focussed (of category *pol*), nor can it host a subject (and thus cannot be of category *foc*). For the time being, we will assign it a new category name, i, and allow seem to select something of this category. We arrive at the following new lexical entries.

Our lexicon can be visualized as per the flowchart in figure 31.

One interesting aspect of our new lexicon is that the resulting language is infinite! This can be seen by inspecting the flowchart, which now contains

Table 23: Lexical entries seem and to

seem :: $\mathtt{i}^{+}.\mathtt{v}^{-}$	to :: $\mathtt{y}^{+}.\mathtt{i}^{-}$

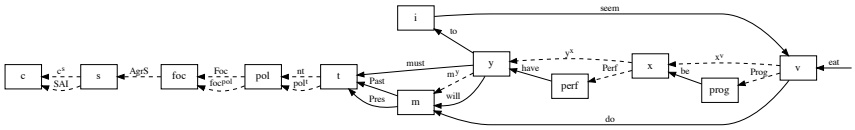


Figure 31: Our grammar after raising to subject

a cycle; we can go from v to y, and then from y back to v (via lexical items to and seem).

2.3.2 Raising to object

We now consider the following sentences.

25. John will expect Mary to laugh.

26. Mary believes John to have been crying.

I assume the dependencies between whole words as shown in figure 32.

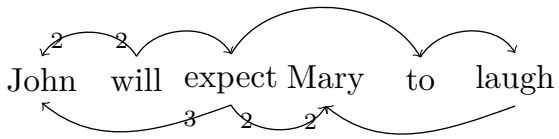


Figure 32: Dependencies for sentence 25

In other words, there is a (selectional) dependency between the embedded verb and the object, as well as one (a case assignment dependency) between the matrix verb expect and this object. There is *also* a (selectional) dependency between the matrix verb and the matrix subject.

Based on these dependencies, we assign the feature bundle $\mathtt{i}^{+}.\mathtt{k}^{+}.{}^{+}\mathtt{d}.\mathtt{v}^{-}$ to the lexical item expect, allowing it to

1. select a non-finite clause ($\mathtt{i}^{+}$)

2. check the case of something in this clause ($\mathtt{k}^{+}$)

3. select its subject ($^{+}\mathtt{d}$)

4. project a vP ($\mathtt{v}^{-}$)

Scouring our lexicon, we see that our other (intransitive) verbs contain a similar feature bundle suffix: $^{+}\mathtt{d}.\mathtt{v}^{-}$. We can decompose our verbs, abstracting out this suffix as a new lexical item: $\mathtt{v} :: \underline{\mathtt{V}^{+}}.{}^{+}\mathtt{d}.\mathtt{v}^{-}$. We can think of this lexical item as akin to the syntactician's argument introducing 'little-v' head. If object case checking is overt, it is crucial that this new lexical item is strong, as the raising to object verb expect appears before the object, yet the object moves to the specifier of expect when its case is checked. Having this 'little-v' head be strong allows expect to head-move to a higher (i.e. lefter) position than the object which moved to its specifier.[25] Our lexical verb lexical items now have the following form:

Table 24: Verbal lexical items

laugh :: V	expect :: $\mathtt{i}^{+}.\mathtt{k}^{+}.\mathtt{V}^{-}$	$\mathtt{v} :: \underline{\mathtt{V}^{+}}.{}^{+}\mathtt{d}.\mathtt{v}^{-}$

Our lexicon can be visualized as per the following flowchart.

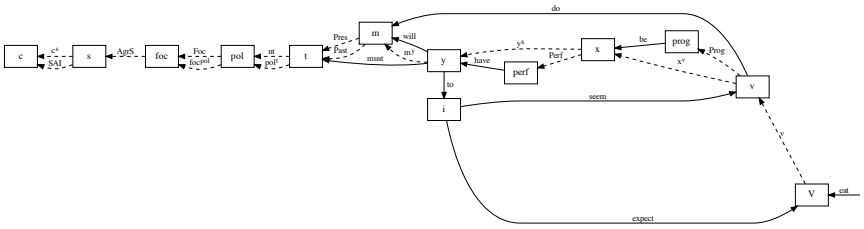


Figure 33: Our grammar after raising to object

[25] Note that if object case checking is covert, then we cannot account for the word order in sentences with auxiliaries! The raised-to-object subject would surface in its base position adjacent to the lower (!) verb, giving rise to ungrammatical sentences like "John expects to have laughed Mary." Our initial dependency structure represented a derivation which did not give rise to the correct word order! This illustrates an alternative perspective on the methodology of this paper; instead of merely reducing redundancy, lexical decomposition can be viewed as an repair operation on *analyses*. We can begin with a whole word analysis capturing the postulated dependencies, without worrying about whether it allows for the correct word order, and then lexical decomposition can be used to obtain an analysis which does.

2.3.3 Passivization

Raising to object verbs become raising to subject verbs when passivized.

27. Mary is expected to laugh

28. John has been being expected to have been crying

I assume the dependencies between whole words as shown in figure 34.

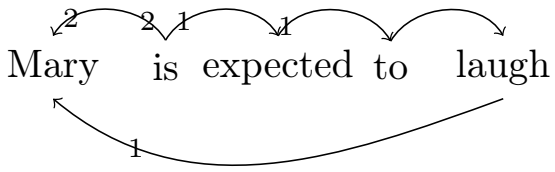


Figure 34: Dependencies for sentence 'Mary is expected to laugh'

Based on these dependencies, we initially assign the feature bundle $\mathtt{i}^+.\mathtt{pass}^-$ to the lexical item expected. We do not want to assign it the same category as seem, because, unlike seem, expected obligatorily combines with a form of be. This be is different from our previous be, as it does not govern the progressive. We assign it the feature bundle $\mathtt{pass}^+.\mathtt{v}^-$. We have expect already in our lexicon, with features $\mathtt{i}^+.\mathtt{k}^+.\mathtt{V}^-$. In order to relate our extant expect and new expected, we can decompose out a common core, expect :: $\mathtt{i}^+.\mathtt{V}_t{}^-$, and are then left with two 'residues': a new 'passive' lexical item Pass :: $\underline{\mathtt{V}_t{}^+}.\mathtt{pass}^-$ and a new 'active' lexical item Act :: $\underline{\mathtt{V}_t^+}.\mathtt{k}^+.\mathtt{V}^-$.

Our lexicon can be visualized as per the following flowchart.

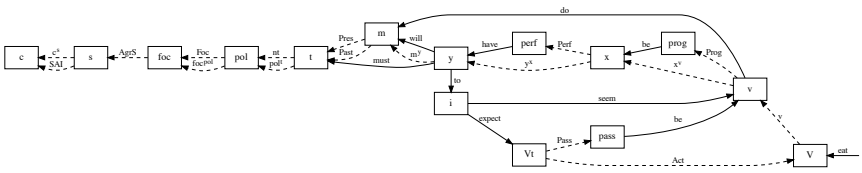


Figure 35: Our grammar after encountering passive raising to object verbs

With this new lexicon, we can immediately analyse transitive verbs (both active and passive):

29. John praised Mary.

30. Mary will have been criticizing John.

We proceed from the following dependencies.

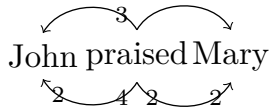


Figure 36: Dependencies for sentence 29

Based on these dependencies, we assign the feature bundle $\mathtt{d}^+.\mathtt{k}^+.\mathtt{d}^+.\mathtt{k}^+.\mathtt{s}^-$ to praised. Decomposing out pieces that already exist in our lexicon, we are left with a single new lexical item praise :: $\mathtt{d}^+.\mathtt{V}_t{}^-$.

3 Conclusion

Minimalist grammars are, I believe, a good formalisation of the program of Minimalism. Instead of trying to dazzle you with the ease with which we can formalize a variety of alternative proposals for mechanisms, ways of feature checking, and so on, I have opted for a more in depth, and hands on approach. My goal was not to convince you that the analysis we arrived at was correct, only that the means of arriving at it are systematic, and straightforward. Even if the grammatical metatheory adopted here is not yours, I believe that the idea of lexical decomposition is generalizable across instantiations of Minimalism, and that decomposition clarifies the nature of functional projections in Minimalism. They are not only justifiable based on *capturing lexical generalizations*, but provide Minimalism's version of type hierarchies. Instead of using individual parse trees to work through the analysis, I opted for the more holistic lexical flowcharts. (The dependency structures are of course equivalent to parse trees for simple whole-word analyses.) This places more of a burden on the reader interested in reconstructing individual parse trees, but I believe that the analyses themselves are familiar enough so that not much is lost. Instead, we can focus on the evolution of the connections between lexical items, which provides a more global perspective from which to understand the development of an analysis.

References

K. Abels. Move? ms. University of Connecticut, Storrs, 2001.

K. Arregi and A. Pietraszko. The ups and downs of head displacement. *Linguistic Inquiry*, to appear.

A. Assmann, D. Georgi, F. Heck, G. Müller, and P. Weisser. Ergatives move too early: on an instance of opacity in syntax. *Syntax*, 18:343–387, 2015.

D. Béchet and A. Dikovsky, editors. *Logical Aspects of Computational Linguistics*, volume 7351 of *Lecture Notes in Computer Science*, Berlin, 2012. Springer.

R. Bernardi and A. Szabolcsi. Optionality, scope and licensing: An application of partially ordered categories. *Journal of Logic, Language and Information*, 17:237–283, 2008.

J. P. Blevins. Periphrasis as syntactic exponence. ms.

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

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

N. Chomsky. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, Massachusetts, 1965.

N. Chomsky. On formalization and formal linguistics. *Natural Language and Linguistic Theory*, 8(1):143–147, 1990.

N. Chomsky. A minimalist program for linguistic theory. In K. Hale and S. J. Keyser, editors, *The View from Building 20*. MIT Press, Cambridge, Massachusetts, 1993.

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

N. Chomsky. Minimalist inquiries: The framework. In R. Martin, D. Michaels, and J. Uriagereka, editors, *Step by Step: Essays on Minimalist Syntax in Honor of Howard Lasnik*, pages 89–155. MIT Press, Cambridge, Massachusetts, 2000.

N. Chomsky. Beyond explanatory adequacy. In A. Belletti, editor, *Structures and Beyond: The Cartography of Syntactic Structures*, volume 3 of *Oxford Studies in Comparative Syntax*, chapter 3. Oxford University Press, 2004.

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

B. Citko. On the nature of merge: External merge, internal merge, and parallel merge. *Linguistic Inquiry*, 36(4):475–496, 2005.

C. Collins. Eliminating labels. In S. D. Epstein and T. D. Seely, editors, *Derivation and Explanation in the Minimalist Program*, pages 42–64. Blackwell, Oxford, 2002.

M. Ermolaeva. Induction of minimalist grammars over morphemes. In A. Ettinger, G. Jarosz, and M. Nelson, editors, *Proceedings of the Society for Computation in Linguistics*, volume 3, pages 484–487. SCiL, 2020.

D. Flickinger. *Lexical Rules in the Hierarchical Lexicon*. PhD thesis, Stanford, 1987.

H.-M. Gärtner. *Generalized Transformations and Beyond: Reflections on Minimalist Syntax*. Akademie Verlag, Berlin, 2002.

T. Graf. Closure properties of minimalist derivation tree languages. In S. Pogodalla and J.-P. Prost, editors, *LACL 2011*, volume 6736 of *Lecture Notes in Artificial Intelligence*, pages 96–111, 2011.

T. Graf. Movement-generalized Minimalist grammars. In Béchet and Dikovsky [2012], pages 58–73.

T. Graf. A computational guide to the dichotomy of features and constraints. *Glossa*, 2:1–36, 2017.

I. Heim and A. Kratzer. *Semantics in Generative Grammar*. Blackwell Publishers, 1998.

R. Hudson. *An Introduction to Word Grammar*, volume 60 of *Cambridge Textbooks in Linguistics*. Cambridge University Press, 2010.

R. Jackendoff. Morphological and semantic regularities in the lexicon. *Language*, 51(3):639–671, 1975.

G. M. Kobele. *Generating Copies: An investigation into structural identity in language and grammar*. PhD thesis, University of California, Los Angeles, 2006.

G. M. Kobele. Minimalist tree languages are closed under intersection with recognizable tree languages. In S. Pogodalla and J.-P. Prost, editors, *LACL 2011*, volume 6736 of *Lecture Notes in Artificial Intelligence*, pages 129–144, 2011.

G. M. Kobele. Importing montagovian dynamics into minimalism. In Béchet and Dikovsky [2012], pages 103–118.

G. M. Kobele. Meeting the boojum. *Theoretical Linguistics*, 40(1-2):165–173, 2014.

G. M. Kobele. LF-copying without LF. *Lingua*, 166, part B:236–259, 2015.

H. Koopman. Recursion restrictions: Where grammars count. In T. Roeper and M. Speas, editors, *Recursion: Complexity in Cognition*, volume 43 of *Studies in Theoretical Psycholinguistics*, chapter 2, pages 17–38. Springer, 2014.

M. Kracht. Syntax in chains. *Linguistics and Philosophy*, 24(4):467–529, 2001.

H. Lasnik. Verbal morphology: *Syntactic Structures* meets the minimalist program. In P. Kempchinsky and H. Campos, editors, *Evolution and Revolution in Linguistic Theory: Essays in Honor of Carlos Otero*. Georgetown University Press, Georgetown, 1995.

J. L. Lee. Automatic morphological alignment and clustering. Technical Report TR-2014-07, Department of Computer Science, University of Chicago, May 2014.

W. D. Meurers. On expressing lexical generalizations in HPSG. *Nordic Journal of Linguistics*, 24(2):161–217, 2001.

J. Nunes. Sideward movement. *Linguistic Inquiry*, 32(2):303–344, 2001.

T. Osborne, M. Putnam, and T. Groß. Bare phrase structure, label-less trees, and specifier-less syntax. is Minimalism becoming a dependency grammar? *The Linguistic Review*, 28:315–364, 2011.

C. J. Pollard and I. A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, 1994.

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

E. Shima. A Preference for Move over Merge. *Linguistic Inquiry*, 31(2): 375–385, 2000.

E. P. Stabler. Derivational minimalism. In C. Retoré, editor, *Logical Aspects of Computational Linguistics*, volume 1328 of *Lecture Notes in Computer Science*, pages 68–95. Springer-Verlag, Berlin, 1997.

E. P. Stabler. Remnant movement and complexity. In G. Bouma, E. Hinrichs, G.-J. M. Kruijff, and R. Oehrle, editors, *Constraints and Resources in Natural Language Syntax and Semantics*, chapter 16, pages 299–326. CSLI Publications, 1999.

E. P. Stabler. Computational perspectives on minimalism. In C. Boeckx, editor, *The Oxford Handbook of Linguistic Minimalism*, Oxford Handbooks in Linguistics, chapter 27, pages 616–641. Oxford University Press, New York, 2011.

P. Svenonius. Spans and words. In D. Siddiqi and H. Harley, editors, *Morphological Metatheory*, volume 229 of *Linguistics Today*, pages 201–222. Johns Benjamins, Amsterdam, 2016.

H. van Riemsdijk. Grafts follow from merge. In M. Frascarelli, editor, *Phases of Interpretation*, volume 91 of *Studies in Generative Grammar*, pages 17–44. Mouton de Gruyter, 2006.

C. Wilder and H.-M. Gärtner. Introduction. In C. Wilder, H.-M. Gärtner, and M. Bierwisch, editors, *The role of economy principles in Linguistic Theory*, pages 1–35. Akademie Verlag, Berlin, 1997.

E. Williams. *Representation Theory*. MIT Press, Cambridge, Massachusetts, 2003.

A. M. Zwicky and G. K. Pullum. Cliticization vs. inflection: English N'T. *Language*, 59(3):502–503, 1983.