Lexical Decomposition

Greg Kobele

June 18, 2018

1 Decomposition of Feature Bundles

Given a lexical entry $\mathtt{w} :: \alpha$, we can divide α up into the following parts:

- the part that occurs *before* the first category feature $\mathtt{x}$
- the first category feature itself $\mathtt{x}$
- the part that occurs *after* the first category feature

In a useful lexical item (i.e. one which can be used in a convergent derivation), there will be exactly one category feature, the precategorial part will consist of some number of positive merge and move features ($\mathtt{=x}$, $\mathtt{x=}$, $\mathtt{+x}$, $\oplus\mathtt{x}$), and the postcategorial part will consist of some number of negative move features ($\mathtt{-x}$).

So let again $\mathrm{W} = \mathtt{w} :: \alpha\beta\mathtt{x}\gamma$ be a lexical item with $\alpha\beta$ the precategorial part of its feature bundle (one or both of α and β may be empty). We can *decompose* W into two lexical items in the following way (I assume that $\mathtt{w}$ is *fresh*; i.e. no other lexical item has a feature of that type):

$\mathtt{w} :: \alpha\mathtt{w}$	$\epsilon :: \mathtt{=w}\beta\mathtt{x}\gamma$

From a linguistic perspective, this amounts to saying that what we used to think of as an XP (remember that W had category $\mathtt{x}$) is really *two* phrases, XP with complement WP. We can view this process in terms of the trees we would construct in figure 1.

Exercises Decompose the following lexical items at the vertical bar

1. ϵ :: =V +k | d= v
2. give :: =d +k | d= V

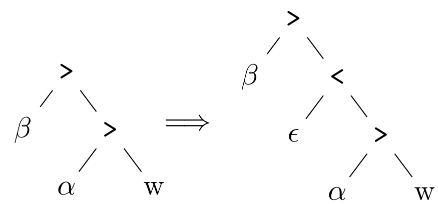

Figure 1: Decomposition in pictures

1.1 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 figure 2. Each of these six lexical items has three features,

will :: `=v.+k.s`	must :: `=v.+k.s`
had :: `=perf.+k.s`	was :: `=prog.+k.s`
has :: `=perf.+k.s`	is :: `=prog.+k.s`

Figure 2: Some tensed auxiliaries

giving us a total lexical size of 18. However, all six lexical items end with the same length two sequence of features: `+k.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, giving rise to the lexicon in figure 3. This lexicon has *seven* lexical

will :: `=v.t`	must :: `=v.t`
had :: `=perf.t`	was :: `=prog.t`
has :: `=perf.t`	is :: `=prog.t`

ϵ :: `=t.+k.s`

Figure 3: Some tensed auxiliaries, with some redundancies factored out

is :: =prog.+k.s been :: =prog.perf

Figure 4: Various forms of (auxiliary) /be/

items, but only fifteen features. We can see that decomposition has *reified* the repeated feature sequence as a new lexical item, which expresses the generalization that subjects move to a specifier position at (or above) TP.

Exercises

1. Identify and decompose redundancies in the following set of lexical items.

John :: d.-k	Mary :: d.-k
the :: =n.d.-k	every :: =n.d.-k
no :: =n.d.-k	some :: =n.d.-k

2. Give a preliminary analysis of the Saxon genitive construction in English (NP's N):
   - John's doctor
   - every man's mother

   Crucially, the NP to the left of the *'s* cannot undergo movement.

1.2 Decomposition and Syntactic Word-Formation

Our decomposition scheme as described above treats the two parts of lexical items (their phonological and syntactic forms) asymmetrically; the syntactic feature sequence is split, but the phonological segment sequence is not. We can imagine, however, wanting 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 figure 4. Not only do both of these lexical items begin with an =prog feature, but (we know) they are all forms of the auxiliary verb *be*. 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 like the following:

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

```
s :: =v.+k.s     en :: =v.perf
be :: =prog.v
```

$$is = be \oplus s$$
$$been = be \oplus en$$

Figure 5: Factoring out /be/

This rule would allow us to split a lexical item into two, but now the original phonological form of the lexical item (w) is factored into two pieces (u and v). The resulting pair of lexical items no longer give us the same structures we would have created with the original one, as now there are two heads (a u and a 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.[1] 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 (`=x`). Second, we record that u and v are jointly pronounced as w.[2] 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 obtain the lexical items in figure 5.

1.3 Complex heads

Once we move from whole word syntax to one which manipulates sub-word parts,[3] we must confront two questions.

[1] Just which head should host the phonological material is something we shall address in a bit.

[2] 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.

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

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

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. 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 three colons separating their phonological and syntactic features, and weak ones with the usual two (as in figure 6).

$$\mathsf{u} ::: \alpha \quad \mathsf{v} :: \beta$$

Figure 6: Strong (left) and weak (right) lexical items

2 Decomposition and Learning

Decomposition can be thought of as a part of a *learning* mechanism for minimalist grammars. In particular, it provides a principled route from a whole-word syntactic analysis to the sort of decompositional syntactic analysis which is characteristic of minimalist-style analyses.

It is known that learning can take place in minimalist grammars in a highly supervised setting [Kobele et al., 2002, Stabler et al., 2003], where

1. words are segmented into morphemes
2. ordered and directed dependencies link words which are in a feature checking relationship
   - the i^{th} dependency of word u connects it to the j^{th} dependency of word v just in case the i^{th} feature of word u was checked by the j^{th} feature of word v
   - the source of the dependency is the attractor feature, and the target of the dependency is the attractee feature

In this setting, the learner is given (essentially) full lexical items where feature names are unique to a particular dependency, and the learner's task is to identify which feature distinctions should be kept, and which should be collapsed. In the cited works (following Kanazawa [1998]), the pressure to collapse distinctions is provided by a limit on the number of homophones in the grammar.

We can use our decomposition mechanism to relax the supervision provided by the segmentation of words into morphemes (1). Accordingly, we assume that we are provided with sentences based on whole words, with dependency links between them as described by point 2 above.

2.1 English auxiliaries

As a simple case study, consider the English auxiliary system. Imagine the learner 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.

The dependencies for sentence 8 are as in 7. From these dependencies, we

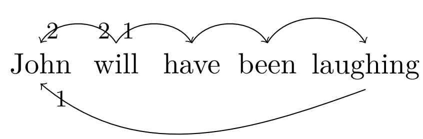

Figure 7: Dependencies for sentence 8

can reconstruct the lexical items in figure 8. Here, the names of the features are arbitrary. After extracting lexical items from sentences 1 – 8, we have

John :: a.-b	will :: =c.+b.s
have :: =d.c	been :: =e.d
laughing :: =a.e	

Figure 8: Lexical items extracted from the dependency structure in figure 7

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*. We can unify these copies into single lexical items by renaming the features involved across the whole lexicon. For example, we might decide to rename the first feature of each of the *John* lexical items to `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 on figure 9.

John :: d.-k	eats :: =d.+k.s
will :: =v.+k.s	eat :: =d.v
has :: =perf.+k.s	eaten :: =d.perf
is :: =prog.+k.s	eating :: =d.prog
have :: =perf.v	be :: =prog.v
been :: =prog.perf	

Figure 9: Lexical items after unification of features

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 a single verb, *eat*, which is appearing in its various forms in this lexicon, this fact is not captured in the grammar. Although this feels right, it is a somewhat wishy-washy argument. A more concrete (although less intuitively appealing) argument to the effect that there are missed generalizations, is that in order to add a new verb to our grammar we would need to add *four* separate lexical items (six, if we had included the past tense and the passive voice), one for each cell in its (derivational) paradigm.

We thus want to express generalizations about our language in terms of our theory, and this we will do via decomposition. There are many ways to begin; we want to compare pairwise lexical items to one another which have similar prefixes/suffixes and (ideally) similar phonologies. We should then decompose, and unify, decompose, and unify, until further decomposition does not achieve any succinctness gains. However we will here simply note *en masse* that the *eat* verbs begin with *eat*, and with the feature `=d`, and decompose them. The result is shown in figure 10 Note that the original

John :: `d.-k`	s :: $\underline{\texttt{=V}}$`.+k.s`
will :: `=v.+k.s`	ϵ :: $\underline{\texttt{=V}}$`.v`
has :: `=perf.+k.s`	en :: $\underline{\texttt{=V}}$`.perf`
is :: `=prog.+k.s`	ing :: $\underline{\texttt{=V}}$`.prog`
have :: `=perf.v`	be :: `=prog.v`
been :: `=prog.perf`	eat :: `=d.V`
$eats = eat \oplus s$	$eaten = eat \oplus en$
$eating = eat \oplus ing$	$eat = eat \oplus \epsilon$

Figure 10: Lexical items after decomposition of *eat*

bare *eat* form has also been decomposed, leaving behind a 'dummy' lexical item which serves to simply change category. This is important, so that no new forms are derived: decomposition does not change the language of the grammar.

In the next step, we do the same with the forms of *be*.[4] This is shown in figure 11. There are three as yet unjustified moves just made:

John :: `d.-k`	s :: $\underline{\texttt{=V}}$`.+k.s`
will :: `=v.+k.s`	ϵ :: $\underline{\texttt{=V}}$`.v`
has :: `=perf.+k.s`	en :: $\underline{\texttt{=V}}$`.perf`
s :: `=x.+k.s`	ing :: $\underline{\texttt{=V}}$`.prog`
have :: `=perf.v`	ϵ :: $\underline{\texttt{=x}}$`.v`
en :: $\underline{\texttt{=x}}$`.perf`	eat :: `=d.V`
	be :: `=prog.x`
$eats = eat \oplus s$	$eaten = eat \oplus en$
$eating = eat \oplus ing$	$eat = eat \oplus \epsilon$
$been = be \oplus en$	$be = be \oplus \epsilon$
$is = be \oplus s$	

Figure 11: Lexical items after decomposition of *be*

1. the two ϵ forms are unifiable, but have not been

[4]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.

2. the two *en* forms are unifiable, but have not been

3. the two *s* forms are unifiable, but have not been

Regarding the first, the ϵ forms serve solely to assert **isa** relationships between categories (every expression of type `V` **is a** expression of type `v`). These must not be unified, as their presence preserves the syntactic distinctions present in the input sentences.[5] There are three basic possibilities for dealing with the two *en* forms:

1. unify `V` and `x`

2. assert that `V` **isa** `x`

3. assert that `x` **isa** `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`. The same reasoning applies to the two *s* forms. Adding this information (as an empty lexical item) to our lexicon gives us the lexicon in figure 12.

John :: d.-k	
will :: =v.+k.s	ϵ :: $\underline{\texttt{=V}}$.x
has :: =perf.+k.s	
s :: $\underline{\texttt{=x}}$.+k.s	ing :: $\underline{\texttt{=V}}$.prog
have :: =perf.v	ϵ :: $\underline{\texttt{=x}}$.v
en :: $\underline{\texttt{=x}}$.perf	eat :: =d.V
	be :: =prog.x
$eats = eat \oplus s$	$eaten = eat \oplus en$
$eating = eat \oplus ing$	$eat = eat \oplus \epsilon$
$been = be \oplus en$	$be = be \oplus \epsilon$
$is = be \oplus s$	

Figure 12: Lexical items after asserting that `V` **isa** `x`

We turn now to *have*, which results in the lexicon in figure 13. Again,

[5]They can be replaced by partial ordering statements of the form $V \leq v$ and $x \leq v$ (see Szabolcsi and Bernardi [2008]).

John :: d.-k	
will :: =v.+k.s	ϵ :: $\underline{\texttt{=V}}$.x
s :: $\underline{\texttt{=y}}$.+k.s	
s :: $\underline{\texttt{=x}}$.+k.s	ing :: $\underline{\texttt{=V}}$.prog
ϵ :: $\underline{\texttt{=y}}$.v	ϵ :: $\underline{\texttt{=x}}$.v
en :: $\underline{\texttt{=x}}$.perf	eat :: =d.V
have :: =perf.y	be :: =prog.x
$eats = eat \oplus s$	$eaten = eat \oplus en$
$eating = eat \oplus ing$	$eat = eat \oplus \epsilon$
$been = be \oplus en$	$be = be \oplus \epsilon$
$is = be \oplus s$	$has = have \oplus s$
$have = have \oplus \epsilon$	

Figure 13: Lexical items after decomposing *have*

decomposition has given rise to two unifiable instances of the morpheme *s*. There are the same three options, and searching for the patterns of connectivity in the lexicon between y and x demonstrate that x can become a y (via the route *be–en–have*) but y cannot become an x. Thus we assume that x **isa** y, as is shown in figure 14.

John :: d.-k	
will :: =v.+k.s	ϵ :: $\underline{\texttt{=V}}$.x
s :: $\underline{\texttt{=y}}$.+k.s	
ϵ :: $\underline{\texttt{=x}}$.y	ing :: $\underline{\texttt{=V}}$.prog
ϵ :: $\underline{\texttt{=y}}$.v	
en :: $\underline{\texttt{=x}}$.perf	eat :: =d.V
have :: =perf.y	be :: =prog.x
$eats = eat \oplus s$	$eaten = eat \oplus en$
$eating = eat \oplus ing$	$eat = eat \oplus \epsilon$
$been = be \oplus en$	$be = be \oplus \epsilon$
$is = be \oplus s$	$has = have \oplus s$
$have = have \oplus \epsilon$	

Figure 14: Lexical items after asserting that x **isa** y

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.

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; open class words such as intransitive verbs contribute just two features to our final lexicon, but 9 features (distributed over four lexical items) to our initial one.

References

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

M. Kanazawa. *Learnable Classes of Categorial Grammars*. CSLI Publications, Stanford University., 1998.

G. M. Kobele, T. Collier, C. Taylor, and E. P. Stabler. Learning mirror theory. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+6),* Venezia, 2002.

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

E. P. Stabler, T. C. Collier, G. M. Kobele, Y. Lee, Y. Lin, J. Riggle, Y. Yao, and C. E. Taylor. The learning and emergence of mildly context sensitive languages. In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler, editors, *Advances in Artificial Life*, volume 2801 of *Lecture Notes in Computer Science*, pages 525–534. Springer, 2003.

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