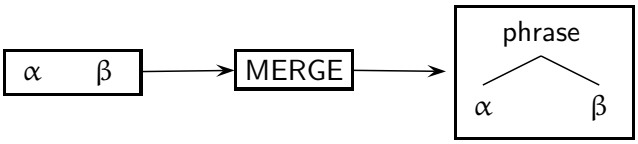
AM2 — Syntax: theories and models
Week 1(2): theoretical basics

The earlier versions of transformational grammar contained many dozens of rules, all of them very specific. Ever since, the trend has been to reduce the number of rules, which consequently forces them to be more general. Nowadays, we assume two core rules, namely (i) *Merge*; and (ii) *Agree*, which we are going to look at in detail in a moment. In order to ensure that very general rules like this can derive the correct range of empirical results, we need to combine them with a much more articulate feature structure on lexical items. In other words, we can state the following generalization on the evolution of the field.

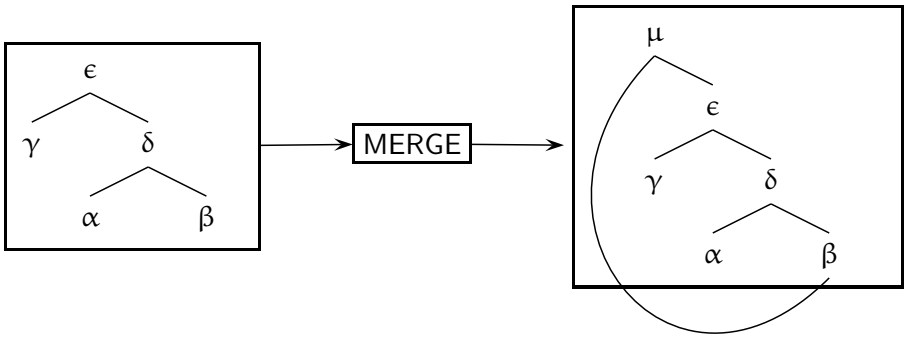
(1) As rules become fewer in number and more general in nature, the featural complexity of lexical items increases accordingly.

1 Basic operations (I): Merge

Merge is the rule that puts two items together to form a larger phrase. A very simple graphic representation is the following, where α and β can be either lexical items or phrases constructed by previous applications of Merge.



There are two subtypes of Merge, which we call *internal* and *external* (technically, there is no difference in the way they operate, but it is descriptively convenient to have a distinction like this). The previous diagram illustrated external Merge, in which neither β nor α are subconstituents of each other prior to Merge. Internal Merge, on the other hand, applies when one of the Merged elements starts as a subconstituent of the other. This is the configuration that is commonly known as *movement*.



1.1 Labelling

Merge doesn't simply put two syntactic objects together: it also provides the resulting phrase with a *label*, which must be identical to the label of either one (but only one) of the two phrases in the input. The point of having a label is to explicitly state the category (N, V, Adj, etc) of the syntactic object in question. We refer to this as the *Projection Principle*: the category (and perhaps other features) of the head are projected (transferred, passed over to) the phrase.

1.2 Subcategorization

What determines whether Merge can apply to two syntactic objects? As with nearly everything else in this system, it is going to depend on features. In order for two syntactic objects to Merge, one of them has to select the other (ignoring adjunction for the moment). Selection is modelled by means of a *subcategorization* feature on the selector, which is simply a statement of the class of categories that the selector can select. For example, a verb like *eat* can have the following feature structure:

(2) *eat* $\begin{bmatrix} [\ \text{category} : \text{verb}\ ] \\ \vdots \\ [\ \text{subcat} : \text{DP/NP}\ ] \end{bmatrix}$

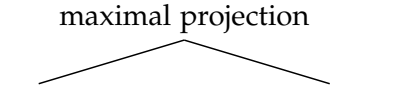
When Merge happens, the [subcat] feature of the selector is satisfied. Certain theories of syntax (e.g., HPSG) would model it as saying that the entire complement of *eat* gets embedded as the value of the [subcat] feature. Within minimalism, some would say that the [subcat] feature is deleted upon satisfaction. We don't need to worry about what happens exactly, so long as we accept that selection is feature-driven.

2 Phrase structure

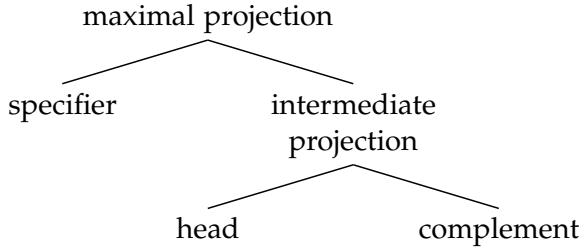
2.1 The parts of the phrase

Merge gives us a way to create large structures out of lexical items —i.e., it provides us with *Phrase Structure*. Here it is useful to have some terms to refer to the parts of a structure. For any one instance of Merge, the selector is going to be called the *head*. The first selected item is going to be the *complement* of the head. If the head selects more than one syntactic object, the second-, third-,...selected syntactic objects are going to be the *specifiers*. The highest projection of the head is going to be a *maximal projection* (notated XP, for any category label X), and the rest are going to be *intermediate projections* (notated X'). Note that, by equating "maximal projection" with "highest projection of a head", we ensure that complements and specifiers are always maximal projections.

(3)

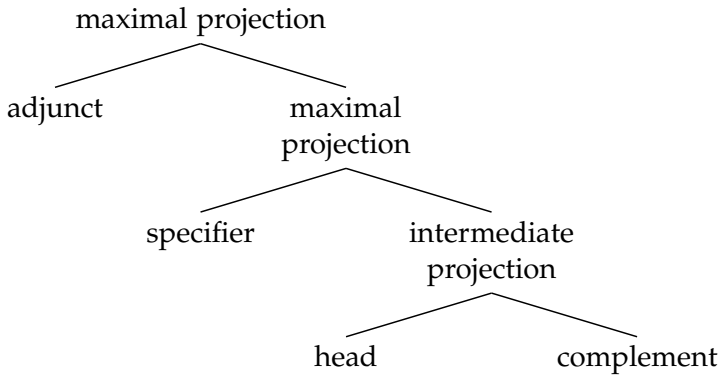


(4)



Beyond specifiers, we have *adjuncts*. Adjuncts differ from specifiers and complements in that they are not selected by the head, and therefore are not required in the structure. Moreover, Merge of an adjunct and a maximal projection doesn't change the maximal projection status of the input; it simply stacks another maximal projection on top.

(5)



You might have noticed that new syntactic objects are always merged at the root (with the exception of Head Movement, which we will get at later). This is called the *Edge Condition*.

(6) *Edge Condition*
Merge always happens at the root.

2.2 Dominance and c-command

With this much in place, we can define two notions that are going to be very widely used during the rest of the course, *viz.*, *dominance* and *c-command*. We start by defining a specific type of dominance, *immediate* dominance.

(7) *Immediate dominance*
Immediate dominance is a reflection of Merge. If $\boxed{\alpha, b} \longrightarrow$ Merge $\longrightarrow \boxed{c}$, then c immediately dominates both α and b. We call c the *mother* of α and b; conversely, α and b are the *daughters* of c.

Early models of transformational grammar stipulated that any given node could have only one mother, i.e., they excluded structures like (xx), where β is immediately dominated by both μ and δ (see, e.g., McCawley 1968, "Concerning the base component of a transformational grammar"). However, this requirement cannot be maintained under the theory of movement that we are going to assume in this course, so we are going to drop this requirement, and allow a node to have an unlimited number of mothers.

(8) *Dominance*
Dominance is the transitive closure of immediate dominance: if α immediately dominates b, and b immediately dominates c, then α dominates c.

To complete the definition of dominance, we need the following two conditions.

(9) *Uniqueness of the root*
There is one and exactly one node, called the *root* of the tree, that dominates every other node and is not dominated by any other node.

(10) *Irreflexivity of dominance (the "no loops" condition)*
If α dominates b, then b doesn't dominate α (this follows from the definition of immediate dominance).

And now we can define c-command on the basis of dominance and immediate dominance.

(11) *C-command*
A node α c-commands a node b iff:
   a. there is a node c such that c immediately dominates α and c dominates b, and
   b. neither α nor b (immediately) dominate the other.

2.3 Movement

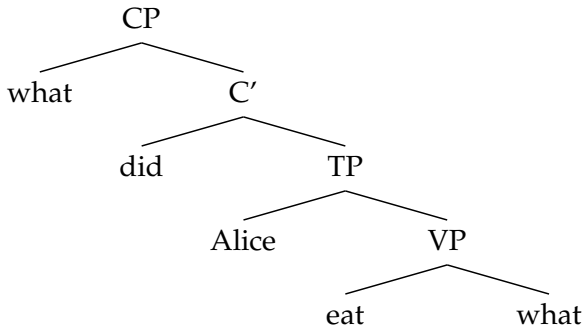
What we call *movement* is a formalization of the intuition that some lexical items are playing a role in more than one part of the structure. For instance, in a question like

(12) What did Alice eat?

what is functioning (i) as a question operator taking scope over the proposition; and (ii) as the object of *eat*. Other theories of syntax formalize this intuition differently. For example, HPSG and Categorial Grammar allow the [subcat] feature of *eat* to be satisfied later than it ought to. In minimalism, though, we want this [subcat] feature to be satisfied *in situ*, so we need to merge *what* as the object of *eat* and then displace it to its surface position.

The implementation of movement has changed through the years. Initially, a moved item only left a *trace* (notated t) in the base position. A trace is simply an indication that there was something there, and that this "something" has moved away (there were independent conditions to ensure that the trace was effectively identified with the moved element). However, later research revealed that the trace has to be more complex than that. In fact, it has to be just as complex as the moved item. This resulted in the *copy theory of movement*, in which the moved item leaves behind an (almost) exact copy of itself. Note that this requires an additional condition stipulating that only one of the two copies (typically, but not always, the highest one) can be pronounced.

(13)



A recent alternative to the copy theory is the *multidominance* theory, in which the moved item is remerged in a higher position. However, various people have argued that, for all practical purposes, the copy and the multidomination theory are actually equivalent to each other. In this course, I am going to use the copy theory just because it makes it easier to draw trees, but if you prefer multidominance, then you can easily redraw all the trees as multidominance trees.

3 Readings

- Additional reading for Oct. 18: Epstein, Samuel. 1998. The derivation of syntactic relations. In Epstein, Groat, Kawashima, and Kitahara (eds.), *A derivational approach to syntactic relations*. Oxford: Oxford University Press.
- Reading for Oct. 22/24: Hale, Ken, and Samuel J. Keyser. 1993. On argument structure and the lexical expression of syntactic relations. In Hale and Keyser (eds.), *The view from building 20*. Cambridge: MIT Press.