CAS LX 522 Syntax I
Spring 2014
Paul Hagstrom

April 22, 2014
Notes at the end of a semester

1 How these notes relate to the previous notes

This is a followup to some notes that were handed out before the midterm. The introduction of the DP has rendered certain parts of the previous notes obsolete, so I have reproduced some of the sections from before here, with DPs in the relevant places. For those sections that are updates to a corresponding section in the previous notes, I will put an asterisk (*) next to section title.

2 Ditransitive verbs and UTAH*

This section has been updated in the following ways: DPs are now used instead of NPs, and the θ-roles for Poss have been added.

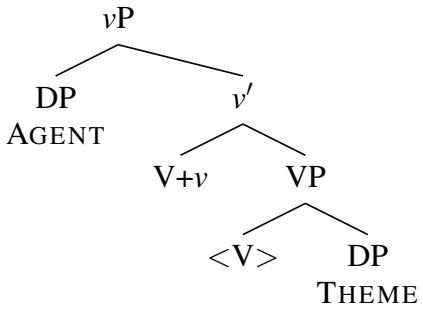
It seems that the largest number of θ-roles any verb can require is three, an example of such a verb being *put*. In order for the UTAH to be able to designate these positions in an unambiguous way, we need to posit a two-level verb phrase. In a sense, verb phrases are always made of two distinct phrases, which we have called the phrase headed by "Big V" (the lower of the two phrases) and "little *v*" (the higher of the two phrases). We can also tell by the word order in sentence like *Mary gave a book to John* that the verb is pronounced in the position of the "little *v*," which we take to be an indication that big V has moved to little *v*.

In the interests of simplicity (in the form of structural uniformity), we assume that verbs always (whether ditransitive or not) have this two-stage structure (both a little *v* and a big V, with little *v* above big V in the structure), and that big V always moves to little *v*. With this much in place, we can now state the UTAH (as it pertains to verbs) more precisely.

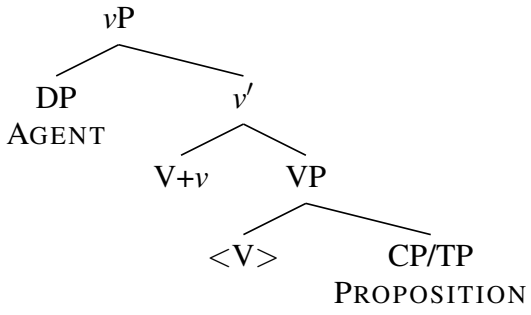
Uniformity of Theta Assignment Hypothesis (UTAH)	
AGENT	DP daughter of v_{agent}P
EXPERIENCER	DP daughter of v_{exp}P
THEME	DP daughter of VP
GOAL	PP sister of V
PROPOSITION	CP/TP sister of V
POSSESSEE	*n*P sister of Poss
POSSESSOR	DP daughter of PossP
EXPERIENCER	PP daughter of VP

There are some aspects of the assignments above that might suggest other, possibly better, analyses. For example, we have two positions that an EXPERIENCER might appear, one for EXPERIENCERs like *(to) Chris* in *Pat seems to Chris to be drunk*, and the other for EXPERIENCERs like *Chris* in *Chris fears spiders*. Yet a third (so far unpredicted) place where EXPERIENCERs can be is for *Chris* in *Spiders scare Chris*, even though here too it seems like *Chris* should be an EXPERIENCER (notice that in *Spiders scare Chris*, there is no visible P associated with *Chris*). So, there are some open questions about exactly how EXPERIENCERs work, although most of the time they act syntactically the same way that AGENTs act.

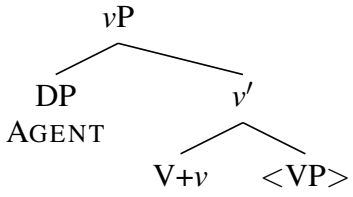
Transitive verb (with THEME)
kick, eat, see



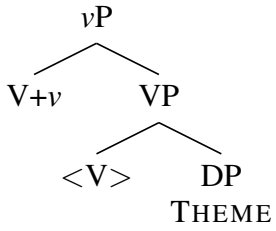
Transitive verb (with PROPOSITION)
say, hear, claim



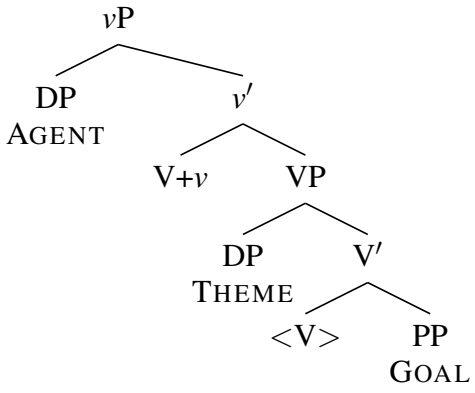
Unergative verb
dance, jog, shout



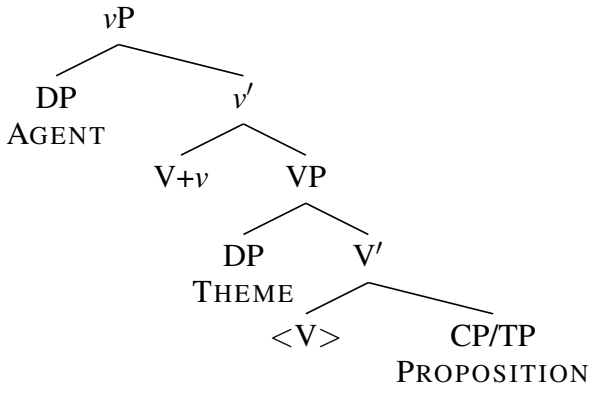
Unaccusative verb
fall, sink, melt



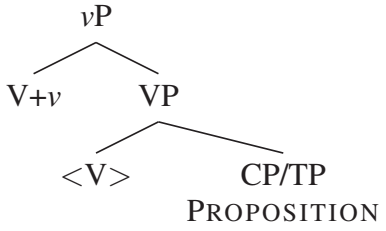
Ditransitive verb (with GOAL)
put, send, give



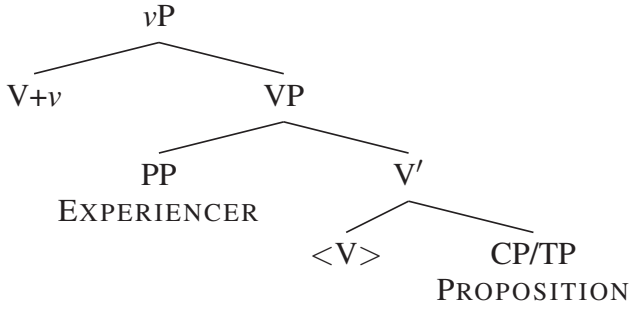
Ditransitive verb (with PROPOSITION)
tell, persuade, promise



Raising verb
seem, appear



Raising verb (with EXPERIENCER)
seem (to Pat)



Thanks to UTAH, we know, for every argument, where it started out. That's important, let me say that again. **Thanks to UTAH, we know, for every argument, where it started out.** So, one of the most important things you can do when trying to determine the structure underlying a sentence you are trying to analyze is work out what the Agents, Themes, Goals, etc., of the verbs involved seem to be.

3 Merge, Move, Adjoin*

This section has been updated in the following ways: DPs are now used instead of NPs, head-movement is discussed, and strong uninterpretable features are now included.

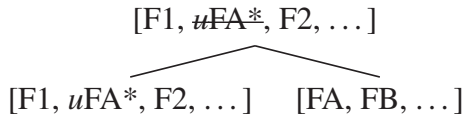
The engine of our model of syntax are the operations that put things together to form larger syntactic object (*Merge* and *Adjoin*), and the way that these contribute to the elimination of the uninterpretable features of the structure being built (*Agree*).

The metaphor we've been using is one of a workbench, where all of the lexical items that are going to be used to form a sentence are scattered about. One by one, we pick them up and combine them with another lexical item (or some already-built object) to make a new object (which we then put back on the table).

When we put two things together (*Merge*), we create a combination that itself has certain properties. The properties that the combined object has seems to be essentially the properties of one of the two things that were combined. So, of two things that are put together by Merge, one of them "projects" its properties to the combined object.

The other thing we assume is that we don't gratuitously or randomly put things together (at least not with Merge). We suppose that we put things together when we *need* to put them together. One of the reasons that we might need to put two objects together is if one object is uninterpretable without the other. (Another reason we might do this is following the Hierarchy of Projections, which will be explored a bit more in the next section).

The determination of which object projects its features to the combination of the two objects seems to be determined by which of the two objects "motivated" the Merge—which of the two objects had a need for the other one. This can be shown very abstractly as follows:



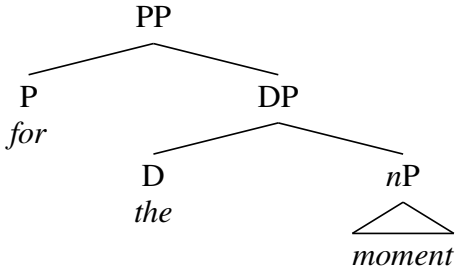
Here, an object that needed the feature [FA] was combined with something that had the feature [FA], and the features of the combined object are the features from the object that needed the feature [FA].

We have adopted some notational conventions and terminology for drawing these (recursively) combined objects into tree structures, based partly on traditional ways of labeling these objects.

A **head** is generally a single lexical item, taken from the lexicon, the "atoms" of the structure, not made up of anything else. A standard way to notate a head in a tree is by writing its category feature (N, V, Adj, P, C, T) and below that its phonological realization and/or some other means of uniquely identifying which lexical item it is, and sometimes below that (some of the) additional features that the lexical item has:

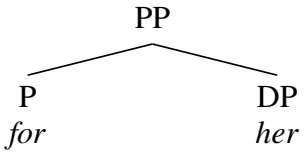
C	P	D
that	*for*	*the*
[clause-type:decl]	[*u*D*, *u*case:acc]	[*u*case:]

When we combine two objects, one of the two *projects* its features to the combined object. The one that *didn't* project its features is—by virtue of not projecting its features—a **maximal projection**. Its features have reached the end of the road. Since its features will not be the features of the combined object, its features will project no further. A maximal projection is often labeled with a P after its category feature (traditionally, the "P" stands for "phrase"). So, below, the features of the D characterize the combination of the D and the *n*P, hence the category of the constituent *the moment* is D. Because it is combined with P and its features do not project further (the whole combined object is of category P), the constituent *the moment* has projected its features as far as it will, and is therefore a maximal projection, written **DP**.

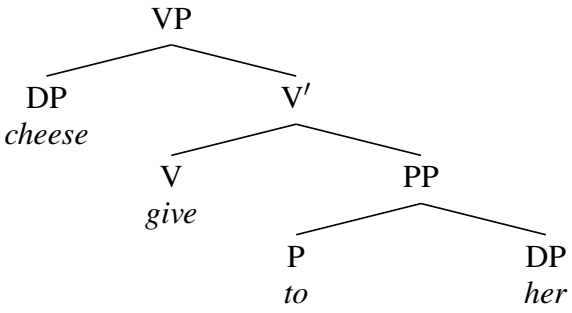


The constituent that is a sister to a head like this is usually called the **complement** of the head, and is—by necessity—a maximal projection. It is standard practice to *always* write the complement as a P, even in light of the comment about heads and maximal projections made immediately below.

Because the determination of whether something is a head and the determination of whether something is a maximal projection are made on the basis of different considerations (something is a head if it comes out of the lexicon as an atom, and is not a combination of any smaller units; something is a maximal projection if its features do not project to the object of which it is a part), it is perfectly possible for a single item to be both a head and a maximal projection. A perfect example of this is the English pronoun. The pronoun *her* has category D, but it also has no needs, it is interpretable (with respect to things it might need to Merge with) as it stands. So, it will not motivate Merge and will not project.



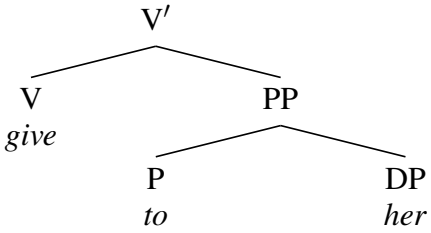
In the tree above, *her* is a head and also a maximal projection. In such situations, it is not clear whether it should be labeled as a head (labeled "D") or as a maximal projection (labeled "DP"). By convention, it is always labeled as DP, but take note that this is a relatively arbitrary decision and plays no role in the function or predictions of the system. Because a head's features can project more than once (a head can have more than one need to be satisfied by Merge), the situation can arise where a constituent is neither a head (it was created by Merge, its features were projected from one of its component objects) nor a maximal projection (because it is subsequently Merged with another object, but its own features project). In such a case, the label is written with a "bar" (which was, in the early days, an actual horizontal bar written over the category label (e.g., $\overline{\text{V}}$), although these days is usually written with a "prime" symbol: V′).



Here, V combines with *to her*, which had category P, and the features of the head V project. The complement's features do not project, so it is a maximal projection of category P, a PP. The constituent *give to her* is then combined with the pronoun *it*, still motivated by a need of the verb, and the features of

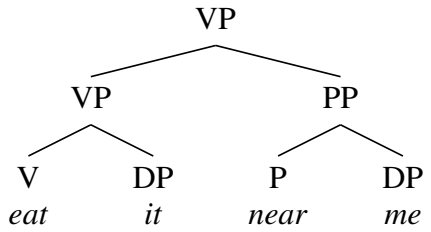
the verb project again to the entire constituent. Thus the whole constituent is the maximal projection (VP), and the give to her constituent is neither a head nor a maximal projection (V′).

The decision about whether to label something as a maximal projection can also be predicted in part by the features it has. If an object has a strong uninterpretable feature (e.g., [*u*D*]) that has not yet been checked, then the object is not a maximal projection—another Merge is necessary, and because this object is motivating the Merge, it's going to project its features to the resulting object. For this reason, when you draw a partial tree, you might still draw an **intermediate projection** (an X′) label even when the tree has not (yet) been extended further:



Merge is a way to combine two objects where the combination is forced (either by one of the objects or by the Hierarchy of Projections). There is a second way to combine two objects, which is not forced: **Adjoin**. In general, we assume that all of the forced operations are carried out first, and only then are the unforced operations allowed. Therefore, adjunction only occurs to maximal projections (although I will say something about head-movement and head-adjunction in a moment—it turns out to be a somewhat different case).

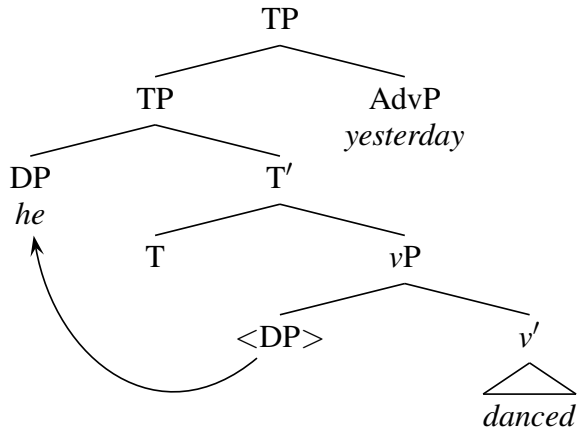
Adjunction is an optional and basically invisible operation as far as the features are concerned. A *v*P is no different whether a PP is adjoined to it or not, a TP with and without an adjunct is exactly the same. As a way to recognize this "invisibility" of adjunction, the standard way of writing adjunction is by simply repeating the label of the object adjoined to as the label of the combined object:[1]



Here, a PP has been combined with a VP, where the VP had already been "finished" and was not going to project its features any further.

Something that is adjoined cannot have any needs that it satisfies by adjoining (at least, not a need of the sort that would have motivated Merge), so something that is adjoined is necessarily a maximal projection—it is not going to project its features further. Again, there is an issue of what label to use when adjoining a head that also happens to be a maximal projection (such as an adverb). By convention, just as with complements discussed above, it is written as being a maximal projection, and—again—this is an arbitrary decision about labeling that doesn't affect the functioning or predictions of the system.

[1]There is at least one piece of evidence for a certain "reality" to this labeling convention for adjunction: The phenomenon of *one*-replacement in English. The word *one* appears to be able to replace any NP in a structure (although, at least in many people's judgments, it cannot substitute for an N that is not an NP). An NP with a PP adjoined to it can be replaced entirely with the word *one*, and the "inner" NP (to which the PP was adjoined) can also be replaced by *one*, leaving the PP adjoined to *one*. That is: *The book of poetry with the red cover on the shelf* can be reduced as *The one on the shelf* or *The one with the red cover on the shelf*.



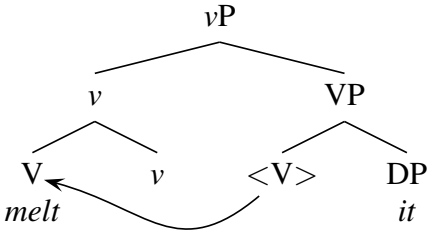
Alongside Merge and Adjoin, it is also often necessary to **Move** something from one part of the structure to another. This is understood as in fact being a process of making a **Copy** of the thing that is to "move" and then incorporating it into the structure a second time, for example Merging it again. When we make a copy of a maximal projection and re-Merge it higher in the structure, the copy is generally labeled by writing the label of the maximal projection that was moved within angled brackets. This was also shown above. It is also very useful (particularly in complicated structures) to **draw an arrow** that indicates what has moved and from where.

You may also notice (above) that, again, the convention of labeling as "XP" objects that are both heads and maximal projections is used with respect to the pronoun *he* in the specifier of TP. The logic here is the same—anything that goes there would necessarily be a maximal projection (although of course it's possible to have things there that aren't heads, e.g., *the children*), so for consistency anything in a specifier (along with anything in a complement, and anything adjoined to a maximal projection) is labeled as a maximal projection even if it also happens to be a head.

The other case of movement that we've seen is **head-movement** where a head moves up to another head. We understand this as a process that creates a "complex head" by in a certain sense fusing the two heads. This is drawn in much the same way as an adjunction structure is drawn, but that obscures a couple of points of difference, so it is worth spending a moment on the characteristics of head-movement.

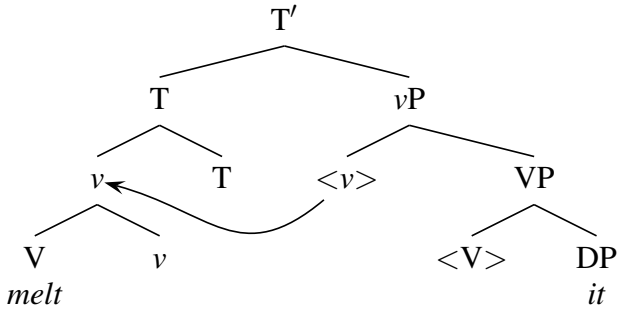
When a head moves up to another head, e.g., V moves to *v*, this doesn't change the fact that the higher head is still the lexical item whose features have projected in the constituent it heads. So, below, the features of *v* are still the features of *v*′, regardless of whether V has moved up to combine with *v* or not. Therefore, the label of the complex head remains *v*. It is for this reason that the combination of two heads via head-movement is thought of as a kind of adjunction: the moving head is attached to the higher head in a way that doesn't alter the essential properties of the higher head.

On the other hand, head-movement, *unlike* XP-adjunction discussed above, *is* motivated by a need of the higher head, in much the same way as Merge is motivated. The higher head is only interpretable when the lower head has moved up to it—that is, an uninterpretable feature is checked by head-movement.



By convention (and in fact in the past people have also proposed this as a substantive aspect of the theory of head-movement), the moving head is attached to the left of the higher head.

Once combined in this way, the complex head is relatively "opaque"—it is still seen by the syntax as an "atom" of sorts, and so if a yet higher head requires a lower complex head to move to it, the entire complex head moves together to form an even-more-complex head higher up. (Note that the structure below would not actually occur in English, because *v* doesn't move to T, but it would in French version of the sentence *it melted*.)



That pretty much covers the types of movement and tree-drawing conventions in our system, so next we can turn to a discussion of motivations for movement and the mechanics of feature checking.

4 Motivations, Agree, feature checking, and the Hierarchy of Projections*

This section has been updated in the following ways: DPs are now used instead of NPs, strong uninterpretable features are now included, and the Hierarchy of Projections has been updated to include D-related projections.

The way that we express the "needs" of a lexical item is by designating the lexical item as uninterpretable unless those needs are met. In the cases already mentioned above, these are situations where, e.g., a preposition needs an object, so we designate P as having an uninterpretable feature [*u*D*], which will be eliminated when an object of category D is Merged with it.

As the system has developed, we have discovered many uses for uninterpretable features in explaining the phenomena in syntax, over an above these "selectional" features.

One could accurately say that the engine that makes this system run is basically the existence of uninterpretable features and the specification of how such features can be eliminated. Through the semester, the notion of feature checking (the elimination of uninterpretable features) has been refined in various ways: First, we considered only checking under sisterhood (essentially, checking by Merge), but further exploration suggested that "long-distance" feature checking is possible, and so "local" feature checking needed to be distinguished from "long-distance" feature checking.

The elimination of uninterpretable features (feature checking) is considered to be a two-step process. First, two features that could participate in checking are identified (and valued, where a value is needed) by the operation **Agree**. Second, if the feature can only be checked "locally" then a movement is effected that would bring the two features close enough together to allow the uninterpretable one(s) to be checked.

Agree can relate any two features that are in a c-command relationship (at least up to phases, covered later). If an uninterpretable feature c-commands a feature that "matches" it (that could check the uninterpretable feature), then those features can be related by Agree. Similarly, if a matching feature c-commands an uninterpretable feature that it matches, those features can be related by Agree. We can informally think of the c-command relation as a requirement for the features to "see each other."

There are two different dimensions on which uninterpretable features can differ from one another. One dimension is the valued–unvalued dimension. An example of a valued uninterpretable feature is the [*u*D*]

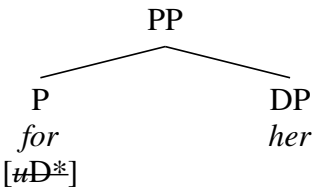
feature on P: this expresses a need for a very specific kind of matching feature, a category feature [D]. An example of an unvalued uninterpretable feature is the [$u\phi$:] feature of T: this expresses a need for some ϕ-features, whatever they may be.

The other dimension is the strong–weak dimension. A weak uninterpretable feature can be checked "at a distance." So long as it "can see" a matching feature, Agree can relate them and the uninterpretable feature is eliminated/checked. A strong uninterpretable feature requires locality—not only must the two features related by Agree be able to "see each other" but they must also be able to "touch each other."

The practical consequence of strong uninterpretable features is that they motivate movement (unless they can be satisfied directly by Merge). Weak uninterpretable features don't motivate movement. We indicate strong uninterpretable features with a "*".

As we find in other parts of the system, Agree is essentially "lazy" and will only relate a feature with another if there was not a closer one. Thus, the [$u\phi$:] feature of T will Agree with the ϕ-features of the Agent and not the Theme in a sentence with a transitive verb (such as *The rhinoceros is extinguishing fires*—it isn't **The rhinoceros are extinguishing fires*).

There are few different "reactions" that we find to a strong feature. Which one is chosen depends partly on the interaction with the Hierarchy of Projections, discussed shortly. If a head has a strong category feature, Merge is one response:



The sisterhood relation between P and DP here is close enough to allow for the strong [*u*D*] feature of P to be checked. This kind of response to a strong feature is the one that is most closely tied to the assignment of θ-roles.

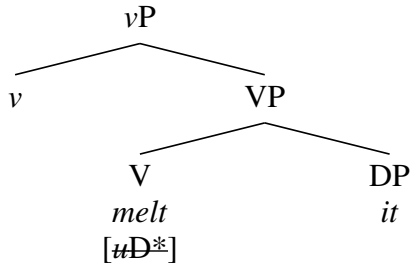
When the Hierarchy of Projections is involved, the Hierarchy of Projections "takes priority." It's not exactly clear why it does, but it nevertheless does. So, let's pause for a moment to review the Hierarchy of Projections.

The **Hierarchy of Projections** ("HoP") is a stipulation that we hope ultimately to be able to derive from the properties of the interpretive component of grammar. It is basically a specification of the order in which lexical heads are Merged into the tree.

Hierarchy of Projections
(C >) T > (Neg >) (M >) (Perf >) (Prog >) (Pass >) *v* > V
D > (Poss >) *n* > N

There are a couple of different things these HoPs tell us. First, many things are optional—if there is no Prog, then it's not a problem. But a couple of things are not optional. T, *v*, V, D, *n*, and N are not optional. We take this to mean that every V must have a *v* above it in the structure, and every N must have an *n* above it in the structure. Every clause must have a T, every DP must have a D. Another point about this is that, at least for D, it's *not* necessary to read *down* the hierarchy. So, a pronoun is just a D and that's fine. Pronouns do not contain *n* or N.

The Hierarchy of Projections is a second way to motivate Merge (the other being the checking of a [strong] uninterpretable category feature). The way this works is that once a constituent is "finished" (there are no strong uninterpretable category features to check, and whatever adjunction there is to be done has been done), the next step is to Merge the finished constituent with the next head up on the HoP.



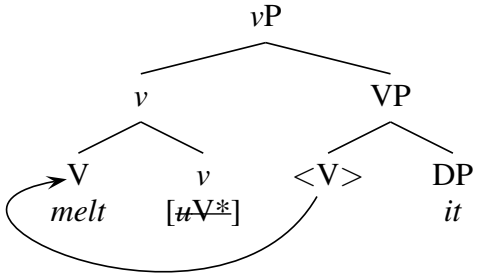
The motivation for Merging *v* with VP above is the Hierarchy of Projections. VP had no further uninterpretable category features to check and was therefore "finished."

One peculiar property of Merging in order to satisfy the HoP is that it creates a configuration in which Agree cannot apply between the two things that Merge. In the tree above, *v* has a [*u*V*] feature (because *v* always does, this motivates the head-movement that we turn to next), but Merging the VP (which has a V category feature) with *v* does not succeed in checking the [*u*V*] feature of *v*, despite the fact that *v* and VP are sisters.

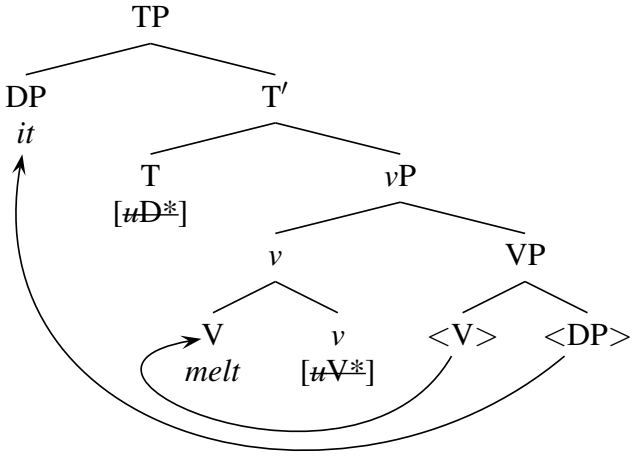
So: Even if there is another reason to Merge, the HoP takes priority, and when Merge happens due to the HoP, the two things Merged are not eligible to check each other's features. (Note, however, that features inside the two things Merged are still eligible for checking. So, T is Merged into the structure as a response to the HoP, but when that happens, it can then see ϕ-features within the *v*P. It just wouldn't be possible for T to check a [*uv**] feature).

Returning now to how the system responds to strong uninterpretable category features, we saw that one response was to simply Merge a constituent with the right category (e.g., satisfying the [*u*D*] feature of P). A second way that such a feature can be checked is by head-movement of a lower head of the right category up to the head that has the strong uninterpretable feature (e.g., checking the [*u*V*] feature of *v*).

Generally, the head-movement option is taken when the sister of the head is of the right category to do the checking, but is prevented from checking due to having been Merged for the HoP. So, as just discussed, *v* and VP are Merged in order to satisfy the HoP and so the [V] feature of VP cannot check the [*u*V*] feature of *v*. In that situation, the lower head moves up to the higher head (which puts the features close enough that the strong feature can be checked):



The final possible response to a strong uninterpretable category feature is movement of a constituent of the appropriate category into the specifier. This option is generally taken when DPs and PPs are involved, and occurs when a potential checking configuration wasn't blocked by a satisfaction of the HoP. An example of this is the movement of the subject into the specifier of TP.



Another situation in which the head-movement option is taken is when an unvalued uninterpretable feature is "valued as strong." These are the cases where auxiliaries move to T, or when T moves to C. When an interrogative C is Merged with a TP (in a main clause, in English), the [clause-type:Q] feature of C values the [*u*clause-type:] feature of T as strong ([*u*clause-type:Q*]). The now-strong feature of T cannot be checked in place, not so much because TP and C are not close enough, but rather because it only became strong "too late" (it wasn't strong initially, and it wouldn't have been strong if C had valued it as [*u*clause-type:decl]). In that case, the head moves up: T moves to C.

5 Functional structure*

This section has been updated in the following ways: D and Pass have been added.

It is possible to divide the heads in a sentence's structure into "lexical" and "functional" heads (and we have many more functional heads than lexical heads). The lexical heads are those that contribute a kind of substantive meaning (nouns, verbs, adjectives, adverbs, possibly prepositions), and the functional heads are the others. The functional heads in a structure each has an important role to play.

- C is considered to be where the clause type is determined—it distinguishes questions from statements.
- T is where tense information is, determining whether a clause is infinitive, past, or present. T also plays a kind of facilitating role in agreement between the subject and the highest verbal element. Finally, T has the "EPP" property (encoded as a [*u*D*] feature on T) that forces all sentences to have a subject.
- *v* is the means for assigning Agent and Experiencer θ-roles, and—given that it is required for every verb—may also be what determines that a verb *is* a verb. There have been proposals, for example, that suggest that the "big V" isn't really a verb at all, but just a lexical root that becomes a verb by virtue of having a *v* above it. This would take us partway toward understanding the relationship between *dance* the verb and *dance* the noun, for example.
- *n* is the counterpart of *v* except for nouns, also responsible for assigning Agent and Experiencer θ-roles.
- D is the uppermost category of the "noun phrase" and is the category that requires case and is often considered to be the interpretive location for concepts like definiteness.

- Poss is a category whose sole purpose is to introduce a Possessor (and assign the θ-role) within a DP.
- Perf marks perfective aspect ("completed").
- Prog marks progressive aspect ("ongoing").
- Pass forms passive verbs, which it does by imposing a requirement on its *v*P complement that the *v* not assign any θ-roles (Agent or Experiencer) nor check accusative case (essentially, requiring that its complement *v*P be the same *v*P that appears in unaccusative structures).
- M are modals, including *shall*, *should*, *will*, *would*, *can*, *could*. The infinitive marker in English (*to*) is also considered to be a modal.

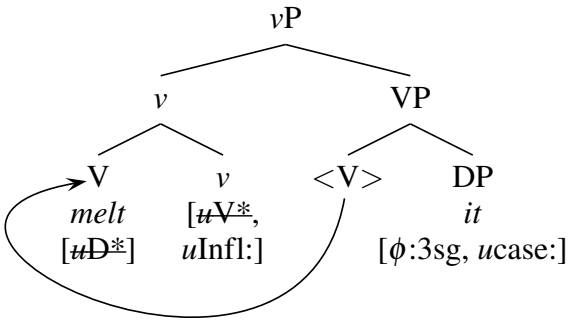
6 Agreement, inflection, and the auxiliary system

It is probably worth taking a special look at the auxiliary system in English and the way in which subject-verb agreement and other inflection is assigned.

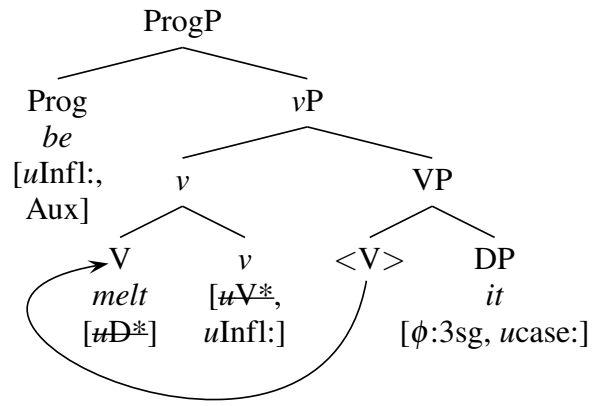
There is a set of things that take inflection, these are things that (at least abstractly) can be inflected. All of the auxiliaries (Perfective *have*, Progressive *be*, Passive *be*, Modals) and the verb (*v*) need to be inflected, which is reflected by having a [*u*Infl:] feature.

A [*u*Infl:] feature is an unvalued, uninterpretable feature that must get a value from something else. Because we build the trees up from the bottom, we generally introduce something with a [*u*Infl:] feature first, and then, later, add something to the tree that will value the [*u*Infl:] feature. That is, [*u*Infl:] features are valued by the next thing up the tree that is capable of assigning inflection.

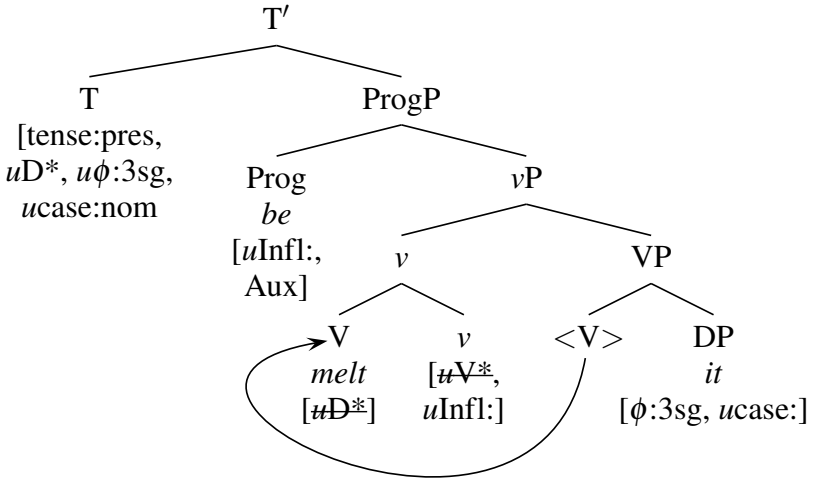
The things that are capable of assigning inflection are T, M, Perf, Prog, and Pass. T is a somewhat special case, so we'll come back to that. For the others, at the point they are Merged into the tree, they can value (and check) the closest [*u*Infl:] feature they c-command. For a simple example with a progressive auxiliary (*It is melting*), we first have the following structure (where I am now explicitly indicating the [*u*Infl:] feature of *v* as well as the ϕ-features and case feature of the DP and the selectional feature of the V *melt*):



Assuming that there is a Prog in the numeration (that is, "on the workbench"), the HoP would have us add that next, resulting in the structure below. Once Prog has been Merged with the *v*P, Prog (or, more specifically, its category feature) will c-command the [*u*Infl:] feature of *v* and will be able to value the [*u*Infl:] (via Agree). Prog itself needs inflection, so it has a [*u*Infl:] feature of its own, but we have now taken care of the inflection on *v* (which ultimately will cause the verb to be pronounced as *melting*).



Suppose now that we have reached the point where we merge T with the ProgP. T has a [*u*ϕ:] feature, and is capable of assigning inflection. When T is Merged, it must first take care of its own [*u*ϕ:] feature. It will c-command the DP *it*, which has ϕ-features, and so the [ϕ:3sg] feature of *it* can value and check the [*u*ϕ:] feature of T. At the same time, the [*u*case:] feature of the *it* can be checked by the [*u*case:nom] feature of T. Then, T (now with ϕ-features valued) can value the [*u*Infl:] feature of Prog, which it does with both its tense value and its ϕ-features. Furthermore, because Prog is an auxiliary (it has an [Aux] feature, as does Perf, Pass, and M), T values the [*u*Infl:] "as strong": [*u*Infl:pres3sg*].



The next step will be to move Prog to T (by head-movement) in order to check the [*u*Infl:pres3sg*] feature of Prog (which, because it is strong, has to be checked very close to T—i.e. with the Prog head-adjoined to T). This happens in the same way that V moves to *v*, and then the derivation continues, moving it into the specifier of TP, etc.

7 DPs and Case

Every D starts off with a [*u*case:] feature—in order to be interpretable, it must have this feature valued and checked. The function of case in a sentence can be thought of as being an indication of where in the structure the DP is—based on what values its [*u*case:] feature.

Nominative case (*I*, *he*, *she*, *they*) goes to DPs that wind up in the specifier of a (finite) T. T has a [*u*case:nom] feature that will match, value, and check on the closest DP to T (which would be the Agent, if there is an Agent).

Genitive case (*my*, *his*, *her*, *their*, *the dog's*) goes to DPs that wind up in the specifier of the silent D that assigns genitive case ($\emptyset_{GEN}$). The $\emptyset_{GEN}$ D has a [*u*case:gen] feature that will match, value, and check

a DP within the complement of $\emptyset_{GEN}$. Note that, since $\emptyset_{GEN}$ is a D itself, it also has an unvalued [*u*case:] feature that needs to get valued and checked elsewhere. So, in *My dog left*, the DP my has its case checked (genitive) by the $\emptyset_{GEN}$ that heads the DP *my dog*, and the DP *my dog* has its case checked (nominative) by the finite T.

Accusative case (*me*, *him*, *her*, *them*) goes to DPs that generally stay in place. Prepositions and *v* can have [*u*case:acc] features that can value such DPs. Whether *v* has a [*u*case:acc] feature depends on whether it assigns a θ-role to a DP in its specifier ("Burzio's Generalization"); the *v* in a passive, and the *v* for an unaccusative, do not have this [*u*case:acc] feature and cannot check accusative case.

Of-case is a special case that goes to DPs that are Themes within an *n*P. It is the direct analog of accusative case inside *v*Ps. The *n* has a [*u*case:of]. The *of*-case is pronounced on a DP by a prefixed *of*, but this of is not part of the syntactic representation, it is just how you pronounce a DP with *of*-case. This means that in *John's gift of cheese to Mary*, *of cheese* is a DP (not a PP).

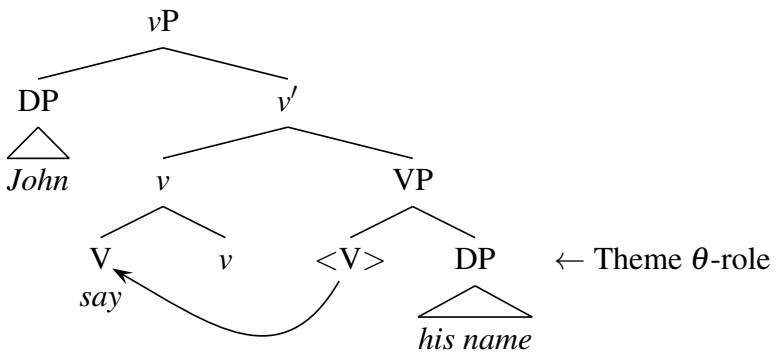
There is also one special instance in which accusative case can be assigned by a V, which arises for Vs that have a possessive have-type meaning. This is specifically required for double object constructions: *John gave me a book*. Here, *me* gets accusative case from *v*, and *a book* gets accusative case from V.

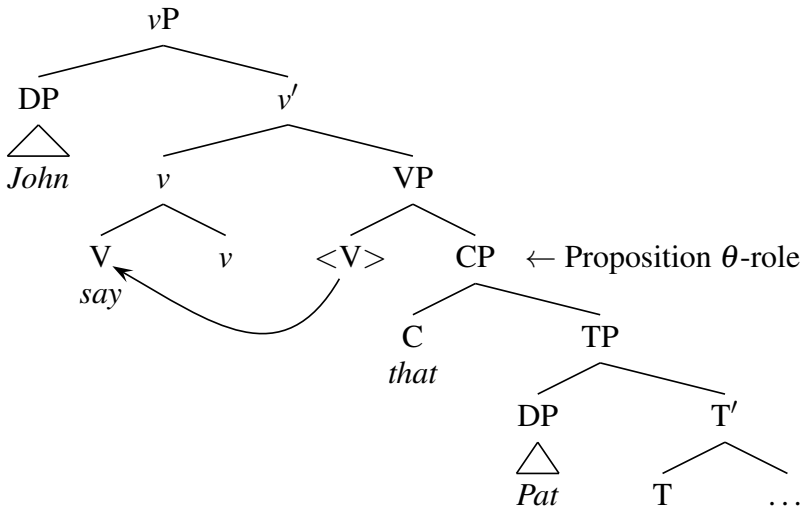
Null case is a special case that can only be assigned to PRO (PRO is a DP and so it needs case too, but it can only get null case—no other case will do). There are two things that can assign null case. One is a silent C ($\emptyset_{NULL}$), and one is the "ing" T, which assigns null case in the same way finite T assigns nominative case. An example of $\emptyset_{NULL}$ would be found in *John wants [$\emptyset_{NULL}$ PRO to leave]*. An example of the "ing" T would be *[After PRO leaving], John went to the store*.

8 Embedded clauses, finite and nonfinite

It is a basic fact of language that sentences can contain other sentences. This can happen in a couple of different ways, but the most basic situation is to have a sentence embedded under a verb like *say* or *want*.

For example, you can have a sentence *Pat ate the bagel* and embed it within another sentence, *John said* [*that Pat ate the bagel*]. The internal sentence (*that Pat ate the bagel*) is treated in the same way a simple object would be (in, e.g., *John said his name*), as complement to the V *say*. When used to embed a sentence, the V assigns a Proposition θ-role to its complement. (In terms of the UTAH, the phrase getting the Proposition θ-role appears as either a TP or a CP, as a sister to V).





It is possible for an embedded sentence to be *nonfinite* as well. The basic case of this is a sentence that has *to* in it before the verb, as in *I want Pat to eat the bagel*. The distinction seems to play an important role in the behavior of sentences in a couple of different respects. A sentence is finite when it has tense and/or person agreement expressed on the verb (*wrote*, *writes*). We find that the C *that* only occurs with finite clauses (whereas *for* occurs with nonfinite clauses: *I said that Pat ate the bagel* vs. *I want for Pat to eat the bagel*). Finiteness is a property that ultimately "lives in" T—it is properties of T that determine whether a clause is finite or nonfinite. To reflect this, we can consider nonfinite T to have the feature [inf] (the idea being that it is short for "infinitive"), while finite T has features like [tense:past] or [tense:pres]. Since only finite T triggers agreement, we can also assume that only finite T has a [$u\phi$:] feature picking up the ϕ-features of the subject. Any T, whether finite or nonfinite, is assumed to have the [uD*] feature (the "EPP feature") that triggers the movement of a DP into "subject position" (the specifier of TP).

When a T is finite, it can trigger movement of a certain class of heads, although the specifics can vary across languages. In English, there is a class of heads we can label "auxiliaries" (including Pass *be*, Prog *be*, Perf *have*, and modals *should*, *could*, *can*, *might*, etc.) that will move to T, although verbs (v) will not move to T. We can distinguish the auxiliaries from the non-auxiliaries by supposing that auxiliaries have a feature that marks them as auxiliaries (the feature [Aux]) and we can say (or, rather, stipulate) that when the [uInfl:] feature of an [Aux]-featured head is valued by finite T, it is "valued as strong"—meaning that although T can set the value of the auxiliary's [uInfl:] feature, it cannot actually check the feature unless the head is right next to T. So, the auxiliary moves to T.

When a T is nonfinite, however, this movement does not happen. We assume that T still values the next [uInfl:] feature down, but it will value it as Inf, and this will be sufficient to check the [uInfl:] feature. (The fact that nonfinite T also differs from finite T in not having a [$u\phi$:] feature could perhaps lead us to make a connection between subject agreement and the need to move a lower auxiliary—but we have not made that explicit anywhere.) The main point to remember here: The word *to* (which we treat as a modal, in the same category as *might*, *could*, *should*) does not move to T.

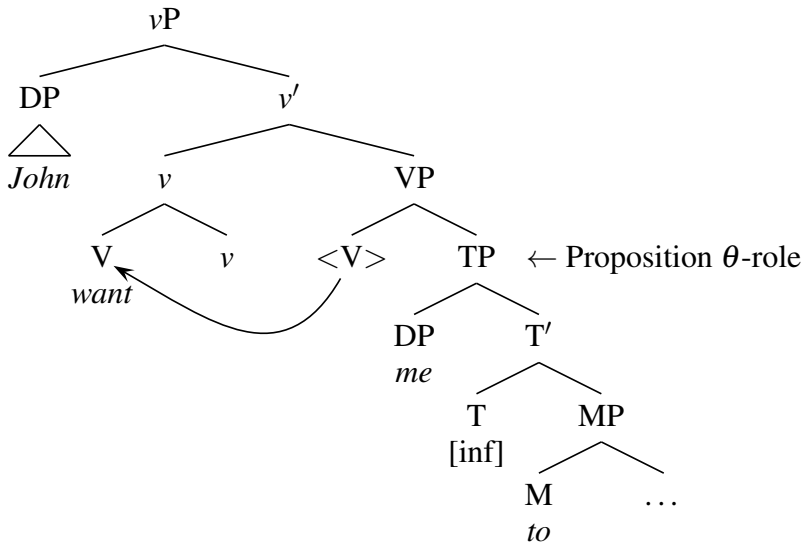
To the extent that languages show verb movement to T, it appears to be basically a universal property of languages that movement occurs to T only when T is finite.

Another difference between finite T and nonfinite T is this: Finite T has a [$u$case:nom] (a.k.a. "[nom]") feature, meaning that it checks and values the [$u$case:] feature (found on every D) as nominative. Nonfinite T in general does not have such a feature and does not check case at all. Thus, even when a DP moves into the specifier of a nonfinite TP (to check the EPP feature), the case needs of said DP have not been resolved. In such cases, the DP might get case from elsewhere (for example, the C *for*, or a higher ECM

verb, discussed below), or it might move on to another SpecTP (as with a raising verb, also discussed below) to get its case feature checked. One exception to the generalization that nonfinite T leaves case features unchecked is the *ing* T found in clauses like *before watching TV*, which has a [*u*case:null] feature, allowing the PRO subject (*before PRO watching TV*).

When T is finite, it is always contained within a CP. Sometimes the C itself is not pronounced, but in those cases it generally could be: *John said (that) Mary stole his bagel*.

When T is nonfinite, the question of whether there is a CP just above it is not as straightforward. In general, there is a CP if there needs to be for some other reason, but not otherwise. Examples of where a C would be needed in a nonfinite clause are: 1) when you can actually see/hear it (*Pat wants for Tracy to leave* or *while eating a bagel*), or 2) when the subject of the nonfinite TP is PRO (*Pat wants PRO to leave*). Particularly in cases where the subject of the nonfinite TP gets accusative case from a higher *v* (ECM constructions such as *John wants me to mow the lawn*), there is no CP above TP.

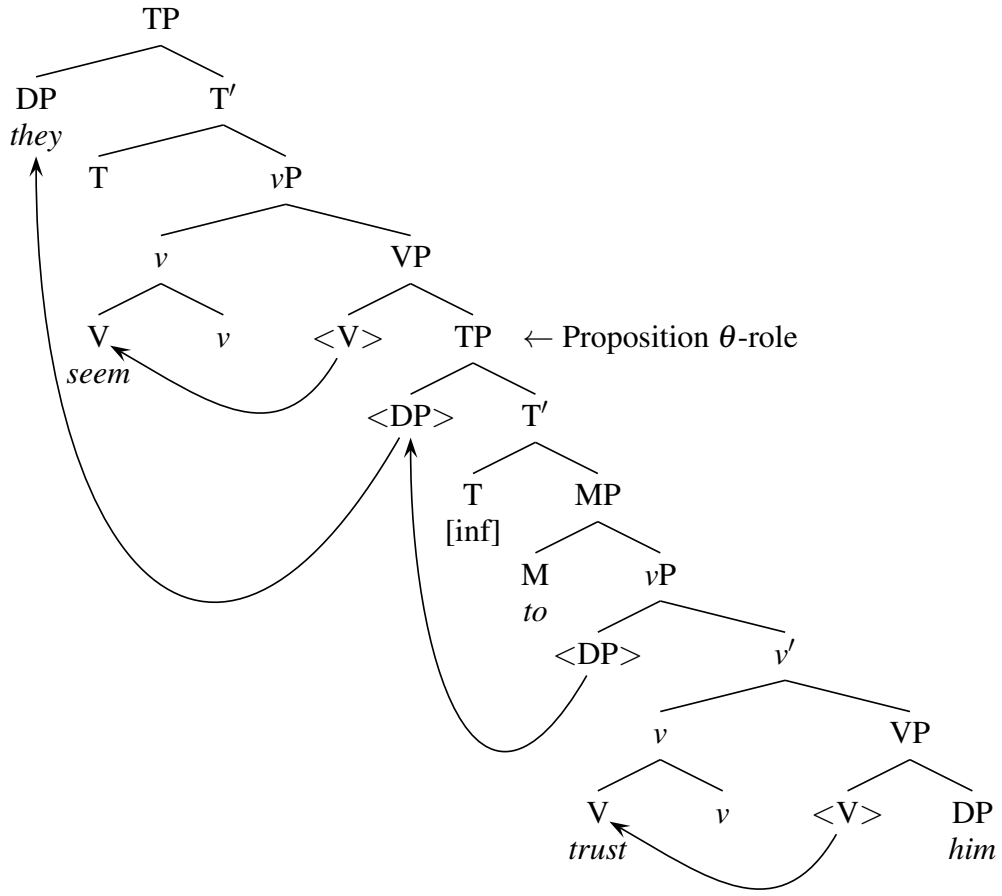


There are a couple of different situations in which you would have an embedded infinitive, and these generally depend on the properties of the verb that embeds it (that is, the verb that provides the Proposition θ-role).

An *ECM verb* ("exceptional case marking") is generally a verb that can also take DP objects to which accusative case is assigned, but can also take propositional complements. For example, *want* or *find* or *believe*. This was illustrated above. *Consider* can take an accusative DP complement (*John considered them*) or a full clause (*John considered them to be annoying*). Notice one important fact about *me* in *John wants me to mow the lawn*: *me* is not an argument of the verb *want*, but rather of the verb *mow*. What John wants when he wants me to mow the lawn is nothing specifically about me except insofar as I am doing the mowing that makes the embedded sentence true. That is, *want* is a relation between the wanter (Experiencer) and the wanted effect (Proposition), and that's all (even if initial intuitions might tend to suggest otherwise). (Compare *John wants nobody to ruin the lawn*—this is not something about nobody, it's about not having the lawn ruined.)

A *raising verb* is a verb that has only a single θ-role, a Proposition (or, sometimes also a Theme/Experiencer of a kind, but crucially nothing that originates in the specifier of *v*P). These are verbs like *seem* or *appear* (or adjectives like *likely*). The fact that a raising verb itself will be within a higher TP means that something will need to occupy the specifier of TP (due to the EPP feature of T). Sometimes (when the embedded clause is finite) this role is played by the expletive *it*, as in *It seems that John has mowed the lawn*. But if there is no *it*, and the embedded clause is nonfinite, then the DP that occupies the specifier of the higher

TP has to be recruited from the lower TP. So, in *John seems to have mowed the lawn*, *John* has actually started in the specifier of the *v*P associated with *mow* in the lower clause, moved into the specifier of the lower TP, and then moved again into the specifier of the higher TP.



A *control verb* is a verb that takes an embedded nonfinite complement, where the "understood" subject of the embedded clause appears to get a θ-role from the higher verb as well. An example where this arises is with the verb *persuade*, as in *They persuaded me to leave*. Here, the meaning of the sentence seems such that the Agent of *leave* is *me*. However, considering the higher verb *persuade*, it seems as if it too is providing a θ-role to *me* (the one persuaded, which we treat as a Theme of *persuade*). By hypothesis (the Unique θ-Generalization), this cannot happen—if there are two θ-roles, there must be two things receiving those θ-roles. The solution to this conundrum is to suppose that there are two DPs, but a) one of them is invisible (called "PRO"), and b) the two DPs share the same referent. So, more technically, a control verb is a verb that takes a nonfinite complement that has PRO as its subject.

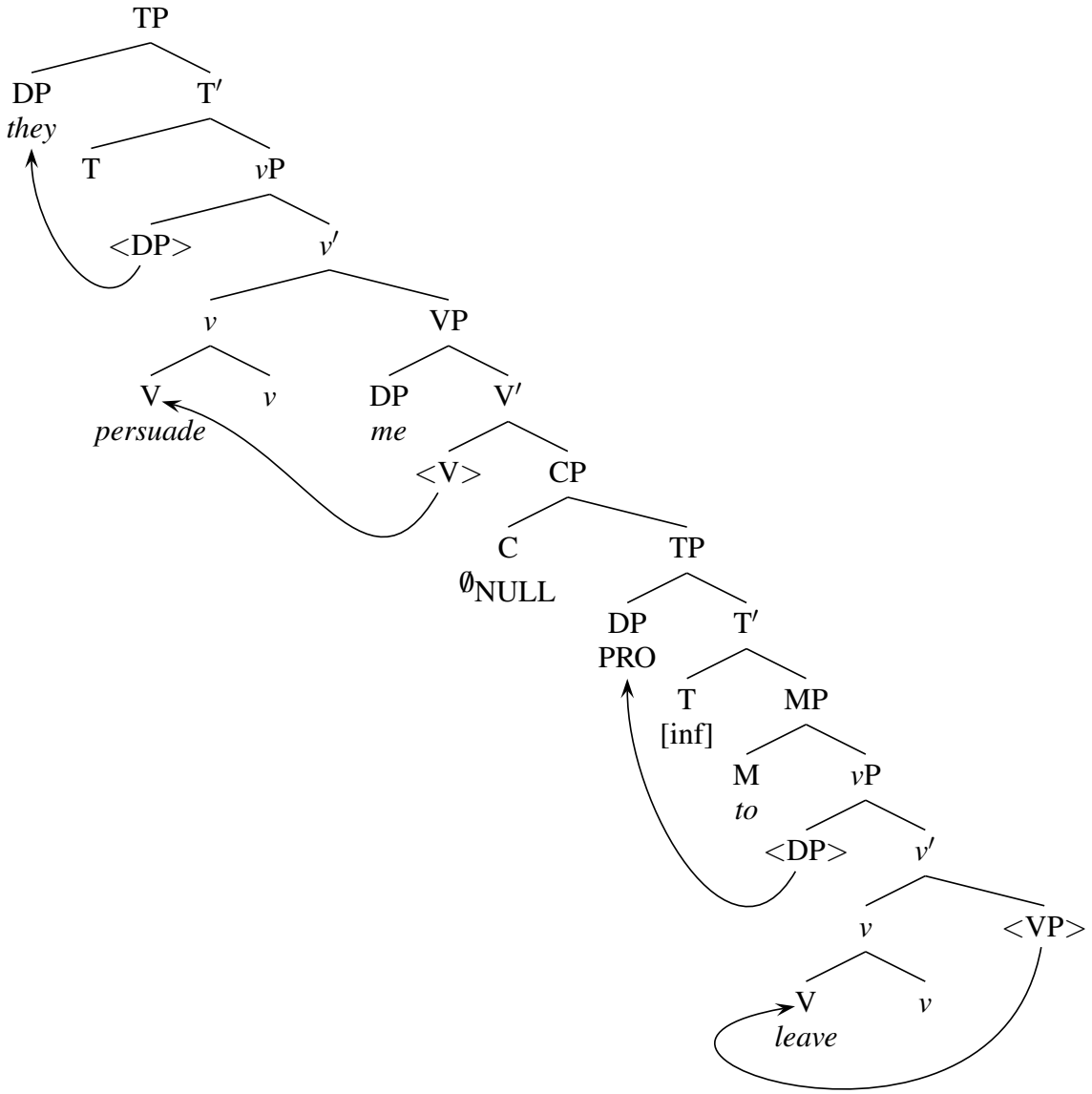
The distinction can perhaps be most easily seen with the verb *want*, which has the interesting property that it can function either as a control verb or as an ECM verb. When *want* functions as an ECM verb, we find the subject of an embedded nonfinite clause surfacing in the accusative case: *John wants me to leave*. When *want* functions as a control verb, no embedded subject is visible—yet, the θ-roles are presumably unchanged from the ECM example: *John wants to leave*. We therefore understand this to be, in fact, *John wants PRO to leave*.

Among control verbs, there are two basic types. Most control verbs are *object control verbs*, so called because the object of the higher verb is necessarily coreferential with the lower PRO. (The word "control" in fact comes from the idea that an argument of the higher verb "controls" the reference of the PRO.) *Persuade* is of this type—the object (Theme) of *persuade* is understood to be the lower Agent (it controls

the reference of PRO): In *John persuaded me PRO to leave*, the object (*me*) the Agent of leave (PRO) necessarily have the same referent.

A small number of control verbs are *subject control verbs*, and this is something that essentially seems to be just an idiosyncratic property of individual verbs (although the inventory of subject and object control verbs across languages is pretty stable, so it is presumably somehow derived from the semantics—but how is not clear). The basic example of a subject control verb is *promise*: If *John promised me to leave* (that is, *John promised me PRO to leave*), it is John that doing the leaving (because John is the subject and, with subject control verbs, the referent of the subject is the same as the referent of the embedded PRO.)

The structure for *They persuaded me to leave* is given below, but it could as easily serve as the structure for *They promised me to leave*, since the difference between the two control verbs is not one that is reflected structurally in the tree. Notice that here, the embedded clause (getting the Proposition θ-role) is a CP: We need the special C $\emptyset_{NULL}$ in order that case can be checked on PRO (recall that PRO requires a special "null" case).



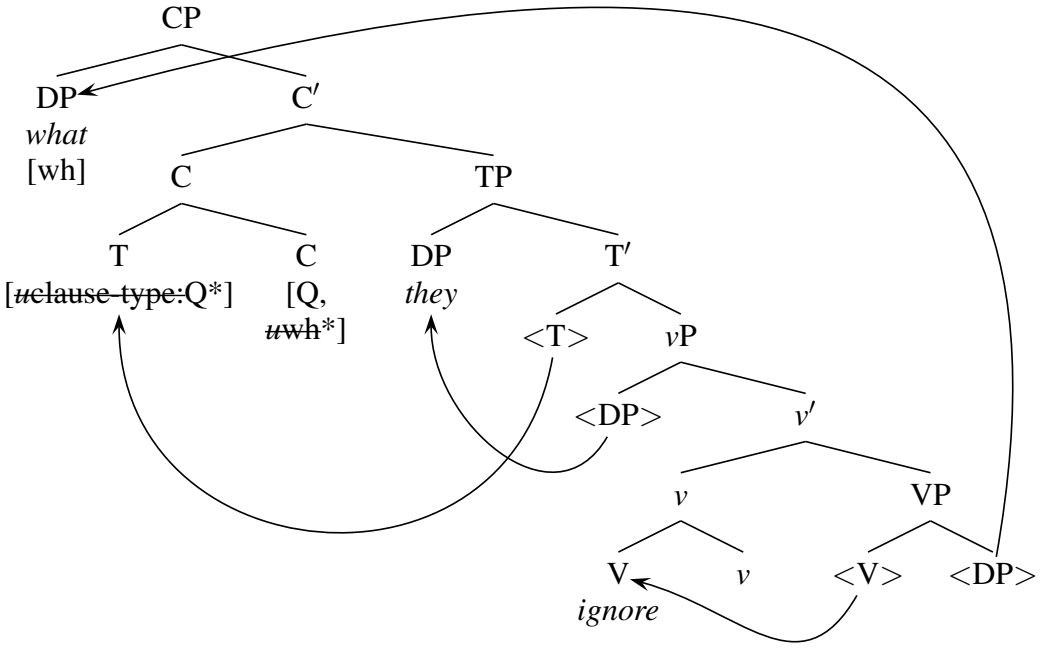
9 Wh-movement and V2…

Many languages seem to require changes in the otherwise normal word orders when information-seeking questions (such as *Who left?*) are formed. This happens in English: In an information-seeking question, the question word seems to need to be first in the clause with which it is associated. A further curious thing happens as well, the subject and "auxiliary" (*be*, *have*, *do*, or modal elements like *can*, *should*, *will*, etc.) switch places. If there was no auxiliary in the corresponding statement, often an extra *do* is thrown in as well.

The terminology used to describe this in our syntactic system arises historically from the study of this phenomenon in English. Question words (often in whatever language) are referred to as *wh*-words (because in English, most such words start with *wh*). The effect of relocating such words to the front of their sentence is called *wh*-movement (movement of *wh*-words), and the inversion of the subject and the auxiliary is often called "subject-auxiliary inversion" (or even just "SAI").

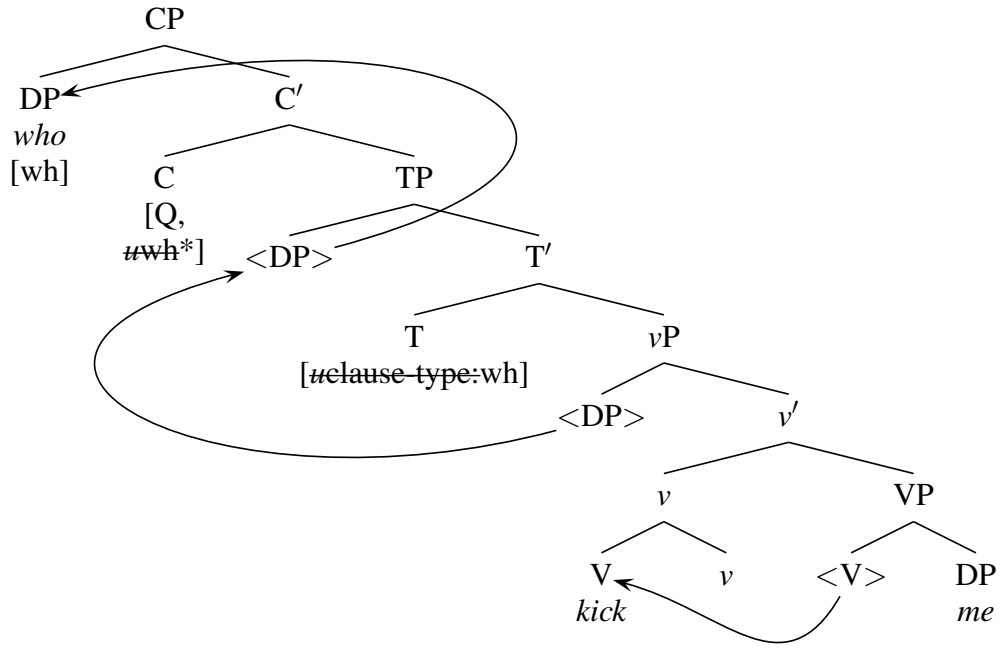
In terms of the system we have developed, there is a quite sensible way to understand what is happening here. The feature that distinguishes questions from statements is a clause type feature that is one of the features of C. If we suppose that interrogative C in some languages (say, English) has a [*uwh**] feature that operates rather like the EPP feature of T, causing something with a [wh] feature to move up into its specifier, we can account for the movement of *wh*-words to the front of the sentence. At the same time, we can suppose that an operation analogous to what causes auxiliaries and/or verbs to move to T (auxiliaries in English, both auxiliaries and verbs in French, etc.) causes T to move to C in main clause interrogatives. Specifically, we can say that T has a feature that is valued by C (a [*u*clause-type:] feature), and when that feature is valued as Q, it is valued as strong (meaning that the feature can only be checked if T moves up to C).

So, *wh*-movement in English involves movement of a phrase with a [wh] feature in its head into the specifier of CP and, in main clause questions, the movement of T to C. (Note that in embedded clauses, this T to C movement does not occur. We simply have to stipulate that only when a main clause C values the [*u*clause-type:] feature of T as [Q] is it valued as strong.)



Subject *wh*-questions are special in that T does not move to C, even in main clauses. We account for this by supposing that a *wh*-word itself can value the [*u*clause-type:] feature of T (thereby keeping C from

valuing it at all, as strong or otherwise), something that only a subject *wh*-word is in a position to do. (Note: that last statement presupposes that only when [wh] c-commands the [uclause-type:] feature of T can [wh] value the [*u*clause- type:] feature.)



Related somewhat to this discussion of *wh*-movement in English, there a number of languages (German, Dutch, for example) that seem to do a very similar thing not just in questions but in all sentences. These are "V2" languages, so called because the (finite) verb appears in "second position" after a topic. We can analyze this as follows: Even declarative C in such languages values the [*u*clause-type:] feature of T as strong, and has a [*u*top*] feature, forcing some phrase (which is marked with a [top] feature) into the specifier of CP.

10 Phases and islands

Movement in this system occurs generally because it has to, in order to check a (strong) feature. It's been observed that, that when something that would have to move is found inside a certain type of constituent, the movement becomes impossible. These constituents that can "trap" a moving element are known as "islands."

The main examples of islands are: Complex Noun Phrase islands, Adjunct islands, and *Wh*-islands. *Wh*-movement is the primary means of detecting these islands—if a *wh*-phrase gets its θ-role inside an island and it has to move to a SpecCP that's outside the island, then the derivation fails and the sentence is ungrammatical.

A complex noun phrase island is any definite DP, like *the book about fish*. You can't ask *What did John read the book about?* and the reason is that *what* would need to move from inside the definite DP out to the SpecCP.

An adjunct island is any adjunct, most obviously clausal adjuncts like *after the President told Congress about the economy*. So, you can't ask *Who did you laugh after the President told about the economy?*.

A *wh*-island is basically any embedded question, like *who John gave the flowers to*. So, you can't ask *What did Mark know who John gave to?*.

The underlying reason for islands existing is taken to be that "movement can't go too far"—and the reason it can't go too far is that the structure is built up in chunks, and once a chunk is finished, you can't see inside it to move something out of it.

The chunks are **phases**, which are CPs. The idea is that once a CP is finished, it is interpreted in some way—basically rendered opaque, except for its "edge" (the specifier of CP).

Because SpecCP is not "frozen" along with the rest of the phase, a *wh*-word that moves into SpecCP can avoid being "frozen" and can remain visible for movement into a higher SpecCP if needed. Observationally, we can see that *wh*-phrases seem to be able to cover arbitrarily large distances (*What did John say Mary heard that Bill bought?*) but there is also evidence that longer movements occur as a result of a number of smaller steps (into each SpecCP along the way). The visibility of SpecCP from outside is also affected by whether the CP itself gets a θ-role (Proposition). If the CP does not get a θ-role, then not even the edge (SpecCP) is visible, and nothing can move out at all. The fact that adjuncts don't get θ-roles then predicts correctly that adjuncts are islands and *wh*-movement should not be possible from within them.

The *wh*-island effect also arises from phases: In an embedded question, SpecCP is already occupied, either by a different *wh*-phrase, or by *Op* (the silent *wh*-word). This means that a *wh*-word that needs to get higher in the tree can't stop in the embedded SpecCP (it is already full), and so it gets caught inside the embedded CP when the phase finishes and the contents of CP are "frozen."

The complex noun phrase island effect suggests that DPs are phases too, or perhaps only definite DPs are phases. Since they have no SpecCP to move to, *wh*-words are trapped within them when the phase finishes. For legitimate cases of "long distance" *wh*-movement, it is assumed that C can (optionally) have a [*u*wh*] feature that will draw the *wh*-phrase up to the intermediate SpecCP position (in order to keep it accessible from above once the phase finishes).