LX 422/722 Intermediate Syntax
Spring 2020
Paul Hagstrom

February 2020
Notes before a midterm

1 What we're trying to accomplish

There are lots of things that we as native speakers know about our native language. We can make very fine-grained judgments on the acceptability and interpretation of sentences in our native language that indicates a relatively complex system of knowledge. We in essence compute (in some way) whether a sentence is grammatical or not, and what interpretation is has. Even given a sentence we've never heard before, we know how it can and can't be arranged and what it would mean.

The theory we've been developing over the semester is an attempt to model and understand the system that underlies these judgments of grammaticality and interpretation. It is an attempt to answer questions like *Why isn't "Students the books read" English? Why can't "she" refer to Mary in "Which book of Mary's did she read?"? How is it that English speakers all know this?*

The basic model we're working with is one in which a set of component parts (words, at least in a certain sense) are provided to a system (a computational procedure of some kind) that assembles them into a structure. Sometimes it will succeed, sometimes (for example, if it is given a set of incompatible words) it will fail. When it succeeds, the structure will come out in a certain (deterministic) way. Our goal is to define a system that is capable of assembling *all*—and *only*—structures for the sentences that we intuitively judge to be grammatical. If the system achieves this, then we have gained a certain degree of understanding of the sort of thing our language knowledge is—that is, we are discovering what it *is* to know a (native) language.

A second consideration is a sort of philosophical one: Children exposed to language data will rapidly come to have the same language system as the adults in their environment. They have reliable success in reaching this point, and many aspects of the system appear to be too complicated to have been deducible from the data the children receive. Given that children seem to "go beyond their input" in this way, we assume that the children must also bring to the task of language acquisition something that constrains or guides the process of language acquisition, some kind of "language instinct" that children have simply by virtue of being human.

In addition to being able to describe, say, the system that underlies a native English speaker's judgments about English sentences, we would like also to understand the nature of this "language instinct." What constraints does it place on the kinds of languages a child can learn, how is it possible that kids can learn so quickly and succeed so reliably?

The hypothesis we have been exploring is that the language instinct, or *Universal Grammar*, consists of the building blocks of a language system—essentially already in place at the point where children are acquiring language—with a narrow range of options that can differ from language to language. The child acquiring a language then has only the task of determining which options are appropriate for the language spoken in their environment. This model of language acquisition (and of syntax) is often referred to as a *Principles and Parameters* model, the idea being that there are a set of basic principles of syntax that are shared by all humans—by virtue of being human—and languages differ on a relatively small number of "settings" or parameters.

We have taken English as our model language for the purpose of exploration and we have looked at it in some depth, but under the Principles and Parameters hypothesis, most of what we learn by looking at English in such detail is equally applicable to other languages as well. Yet, it is still important to consider how things work in other languages too, because this helps us understand what the options are.

The *minimalist program* is a more specific approach to the Principles and Parameters model of language, which takes as its basic hypothesis that syntax is nothing really more than a means of mediating between sound and meaning. Under this view, the mind is understood as segregated into different "modules" responsible for different things. One (at least) module is responsible for understanding, thinking, reasoning, and so forth—the *conceptual-intensional* system. Another module is responsible for running our articulation and perception systems. Each module is separate from syntax, but syntax serves as a point of contact between the modules, generating structures that each module can interpret. Syntax's job is to assemble representations that are in a form that these other systems (the conceptual-intensional and articulatory-perceptual systems) can use. We assume that there is a certain class of things (structures, properties) that these modules are capable of interpreting—those are "interpretable" and everything else is "uninterpretable."

So, the design of our computational system for syntax is one where the building blocks (lexical items, vocabulary items from the language) are assembled in such a way that they are interpretable—nothing that is uninterpretable is left over. If something uninterpretable is left over, the structure cannot be interpreted—we have an ungrammatical structure.

The goal of this model of syntax is to explain the (apparent) complexity we find within and across languages in terms of the procedure for rendering assemblies of lexical items interpretable.

2 The properties of words

When we think about the words from which sentences are formed, we quickly see that they come in different classes. There are major classes of words, those we call nouns and verbs, adjectives and prepositions. Within those classes, there are also subclasses of words, such as the transitive verbs or the intransitive verbs.

So, individual words have properties, like being a verb, or being a transitive verb. This is something we know about the words individually—not necessarily as part of the system, but as part of what we learned when we learned the vocabulary of the language.

As a way for us to indicate the properties of words, we list them as *features*—where "features" is really just a technical term for "property." A noun is said to have the feature [N] or perhaps [category:N] by virtue of being a noun, a verb is said to have the feature [V] or [category:V]. To distinguish transitive verbs from intransitive verbs, we need to rely on a concept of "needing something"—transitive verbs are those that "need" two noun-category things, and intransitive verbs "need" only one. Some verbs don't seem to need any, such as *rain* or *snow*. So, we need to have a way to record the "needs" of a verb.

Moreover, we want to be sure that the needs of the verb are met before the structure is interpreted—we suppose that a transitive verb isn't really interpretable without having both of its nouns. So, we say that a transitive verb has two uninterpretable features, one for each of the nouns it needs, and once we provide the verb with the nouns it needs, then the verb becomes interpretable.

This leads naturally into a formalization of needs and means by which those needs can be satisfied in terms of "uninterpretable features" that need to be "checked" (or removed) before the interpretation system can use them. We write uninterpretable features like [*u*N], meaning that the lexical item that carries this feature cannot be interpreted until it is "given" something that matches the need, in this case something with the category feature [N].

Words have many other properties as well. In many languages, some words seem to carry information about things like plurality (number), semi-arbitrary sub-class membership (gender), and relationship to the discourse participants (person). In order to describe the English pronoun *them*, we need to specify that it is 3rd person (neither speaker nor hearer) and plural.

3 Constituents and the hierarchical structure of sentences

It is readily apparent that sentences have a certain kind of hierarchical structure. A string of words like *the President's book about a goat* turns out to be able to go in most sentences in the same place that a word like *it* can. So, at some level *it* and *the President's book about a goat* are the same kind of thing. On the other hand, *the President's book about a goat* clearly is made up of a bunch of other things. Among them are strings of words that themselves can be replaced by words like *it* or *his* (*The President's book about it*, *His book about it*). We find that there are groups of words in sentences that act together as a unit, which we call *constituents* (parts of a sentence) and that often we can find constituents inside other constituents.

A natural way to view the arrangement of words into hierarchical structures is as a type of (repeated) combination. One word is combined with another to form a combination that can itself be combined with another word (or combination of words), and so forth until the entire structure is built.

Generally, within a *phrase* (a string of words that behaves as a constituent), we find that one word within it seems to serve as its "center," as the most important word in it, that seems to define the properties of the entire phrase. So, the verb phrase *kicked buckets* is at its core the verb *kick*. It seems to be a relatively stable fact about phrases that they have within them such a "center" from which the properties of the phrase as a whole are drawn. One such property is the type of word that the phrase can substitute in for (that is, the category: noun, verb, etc.). So, a "verb phrase" is a phrase whose center, or "head" is a verb. A verb phrase itself acts essentially as if it were a verb, and so we suppose that the head's properties become the properties of the combination of this head with other material. We say then that the features of the head *project* to become the features of the phrase.

From this comes the idea that combinations occur by taking two constituents (each being either a single word or else combination that has already been created) and creating a combination from them, where one of the two things is the *head* and the features of the combination are determined by the features of the head.

4 Verbs, events, and θ-roles

A verb (or predicate more generally) can be thought of as describing a kind of event or state, where different events/states have varying numbers of participants. A verb, by virtue of its meaning, must be marked with some information as to the number of participants its event/state involves. So, *kick* involves two participants, for example, the one doing the kicking, and the one getting kicked. *Put* involves three, the one doing the putting, the one getting put, and the one describing the location where the putting terminates. *Laugh* involves one, the laugher. The participants are said to get *thematic roles*, or technically *θ-roles*. There is one θ-role per participant.

There seem to be a relatively small number of θ-roles, although exactly how many depends a lot on how much you abstract away from the predicates' own meaning. The most basic set of θ-roles appears to include Agent, Experiencer, Theme, Goal, Possessor, and Proposition. We've also posited a Possessee θ-role for the special case of the verb *have*.

Of these θ-roles, Theme is the most general. It is in a broad sense the participant affected by the event/state. An Agent is generally the instigator of the event/state, and is generally a participant to which a mental state and even intent can be ascribed. An Experiencer is consciously affected by an event/state. Because having conscious mental states seems to be part of being an Experiencer or an Agent, a sentence will generally sound anomalous if a non-volitional participant is assigned one of these θ-roles (or be interpreted as if the participant had such mental states, perhaps metaphorically): *The sunlight danced across the surface of the lake*, *The rock kicked John*, *The tree fears lumberjacks*. A Goal generally indicates the termination of a path involved in an event (generally, for events involving motion). A Possessor is a

participant who by virtue of the event/state possesses something, and is generally found as an argument of nouns. Proposition is a role that is assigned to entire clauses, generally something that could in principle be true or false.

The verb/predicate is thought to be basically "in charge" of what θ-roles it assigns. Part of our knowledge of any given verb is the knowledge of which θ-roles it assigns, which participants it "needs." As mentioned earlier, this can be formalized in terms of interpretability: A verb is not interpretable until it has been given as many participants as it needs. We formalize this by putting uninterpretable features on the verb that state the requirements, what would be necessary to render the verb interpretable.

We assume that, in principle, a simpler computational procedure is better (and one can make the case for this on basic principles of scientific investigation generally—the simplest solution is the best solution until shown to be inadequate to deal with the complexity of the data, and the simplest solution is also the one that is most vulnerable to being shown to be inadequate, allowing progress in understanding and predictive power: *Ockham's razor*).

Given the simplicity assumption and the assumption the verbs are uninterpretable until they are provided with their arguments, we hypothesize that what it means to "give" a verb its arguments is to combine the verb with its arguments by means of the procedure (*Merge*) that takes two syntactic objects and forms a composite syntactic object from them, with the features of the composite object being the features of one of the two objects. When a verb that has an uninterpretable feature indicating that it needs an argument is Merged with something that constitutes such an argument, the uninterpretable feature is dispensed with, and, if that is the last one, the verb becomes interpretable (at least with respect to its argument requirements).

This view entails that the satisfaction of θ-role requirements is done *locally*. The arguments of a verb all start off right next to the verb, within the composite object headed by (i.e., that has the features of) the verb. Strictly speaking, it could have been otherwise, but it turns out that as the study of language has progressed over the past 50 years or so, various types of evidence have been unearthed that point to the fact that this locality of θ-role assignment is in fact a property of the system.

Another observation that has been made over that time is that the θ-roles appear to be assigned to quite consistent positions in the structure. Themes are always in more or less the same place, Agents are always in more or less the same place, etc. Post hoc, we can understand why this should be so: for the purpose of learning the words of a language and their meanings, it is quite advantageous to be able to deduce structural information from the basic meaning of the verb. If a verb has an Agent, we know how that is represented in the structure. If we (or a child) hear a new verb *blick* in the following context *John blicked until he got bored and stopped*, we can deduce that John is a Agent, and that therefore John started in the place in the structure where Agents start.

Seeing this tendency, a strong hypothesis was put forward: There is a strict correlation between the θ-role that a participant gets and where it is first introduced into the structure of a sentence. This is the Uniformity of Theta Assignment Hypothesis (or UTAH).

5 Ditransitive verbs and UTAH

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.

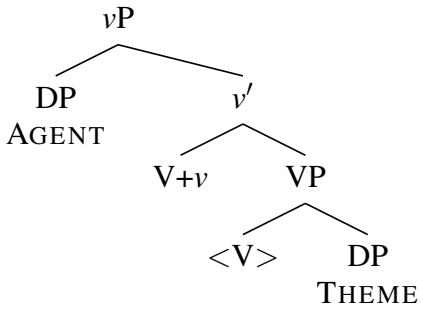
Note: A couple of things we have not talked about yet as of the midterm are included here. This is a more complete statement of the UTAH than you need at the point I gave this out. In particular, anything that refers to *n*P ("little *n*"), N, or NP won't be discussed until after the midterm.

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

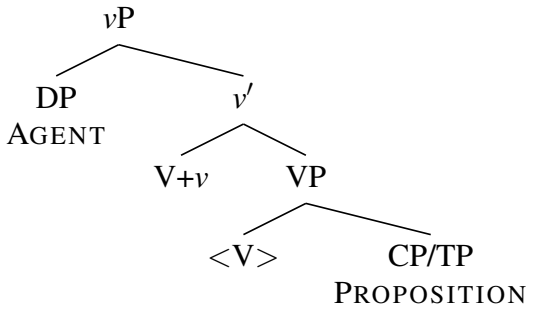
There are some aspects of the assignments above that might suggest other, possibly better, analyses. For example, we have two positions that an EXPERIENCER might appear, one for EXPERIENCERs like *(to) Chris* in *Pat seems to Chris to be drunk*, and the other for EXPERIENCERs like *Chris* in *Chris fears 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. We'll ignore these gaps for now.

Thanks to UTAH, we know, for every argument, where it started out. That's important, let me say that again, in bold: **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.

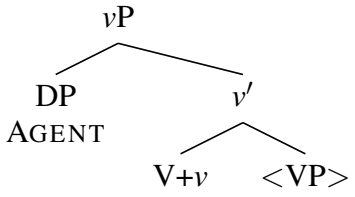
Transitive verb (with THEME)
kick, eat, see



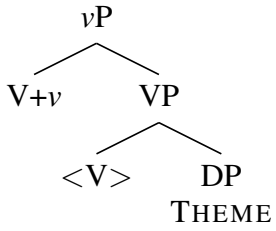
Transitive verb (with PROPOSITION)
say, hear, claim



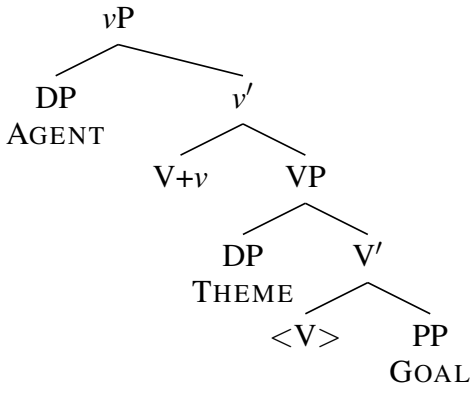
Unergative verb
dance, jog, shout



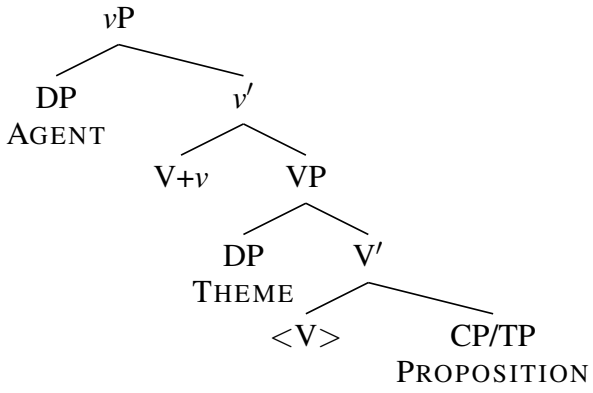
Unaccusative verb
fall, sink, melt



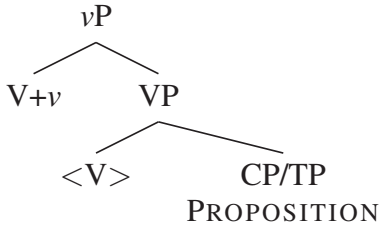
Ditransitive verb (with GOAL)
put, send, give



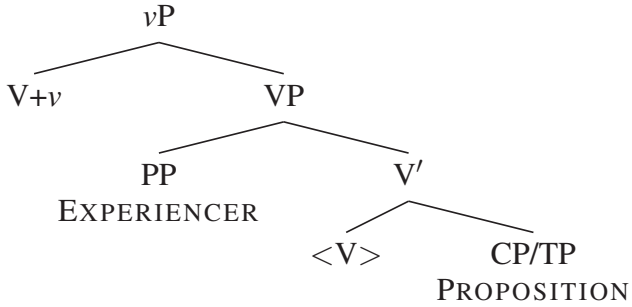
Ditransitive verb (with PROPOSITION)
tell, persuade, promise



Raising verb
seem, appear



Raising verb (with EXPERIENCER)
seem (to Pat)



6 Merge, Move, Adjoin

The engine of our model of syntax are the operations that put things together to form larger syntactic objects (*Merge* and *Adjoin*), and the way that these contribute to the elimination of the uninterpretable

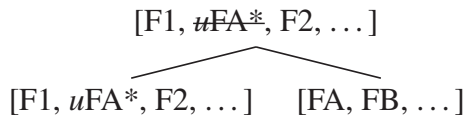
features of the structure being built (*Agree*).

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 workbench).

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:



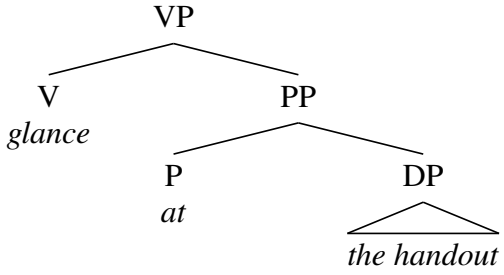
Here, an object that needed the feature [FA] was combined with something that had the feature [FA], and the features of the combined object are the features from the object that needed the feature [FA].

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:

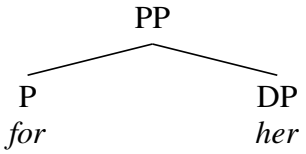
M	P	T
should	*at*	∅
[Aux, *u*Infl:]	[*u*D*]	[tense:past]

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 P characterize the combination of the P and the DP, hence the category of the constituent *at the handout* is P. Because it is combined with V and its features do not project further (the whole combined object is of category V), the constituent *at the handout* has projected its features as far as it will, and is therefore a maximal projection, written **PP**.

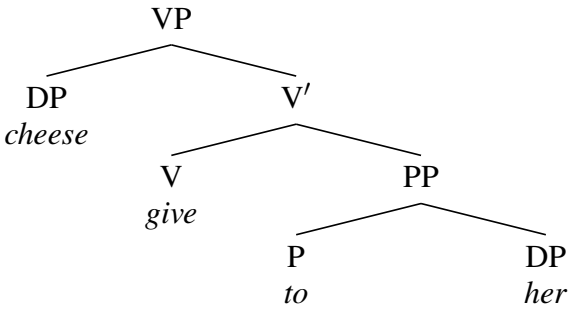


The constituent that is a sister to a head like this is usually called the **complement** of the head, and is—by necessity—a maximal projection. It is standard practice to *always* write the complement as a P, even in light of the comment about heads and maximal projections made immediately below.

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.



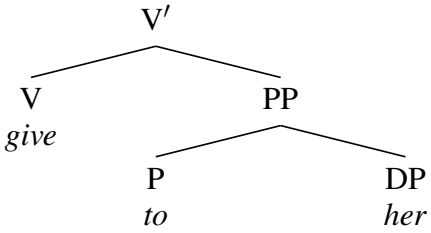
In the tree above, *her* is a head and also a maximal projection. In such situations, it is not clear whether it should be labeled as a head (labeled "D") or as a maximal projection (labeled "DP"). By convention, it is always labeled as DP, but take note that this is a relatively arbitrary decision and 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

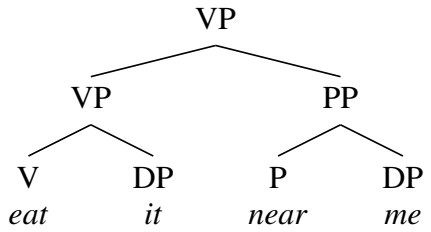
the verb project again to the entire constituent. Thus the whole constituent is the maximal projection (VP), and the give to her constituent is neither a head nor a maximal projection (V′).

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:



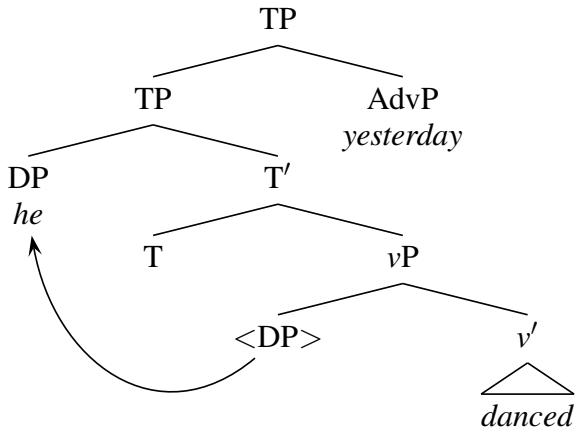
Merge is a way to combine two objects where the combination is forced (either by one of the objects or by the Hierarchy of Projections). There is a second way to combine two objects, which is not forced: **Adjoin**. In general, we assume that all of the forced operations are carried out first, and only 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:



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.



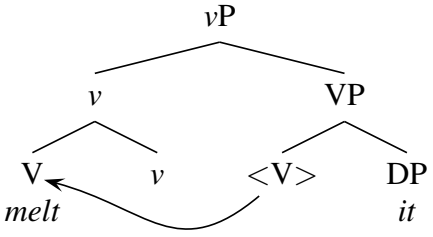
Alongside Merge and Adjoin, it is also often necessary to **Move** something from one part of the structure to another. This is understood as in fact being a process of making a **Copy** of the thing that is to "move" and then incorporating it into the structure a second time, for example Merging it 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.

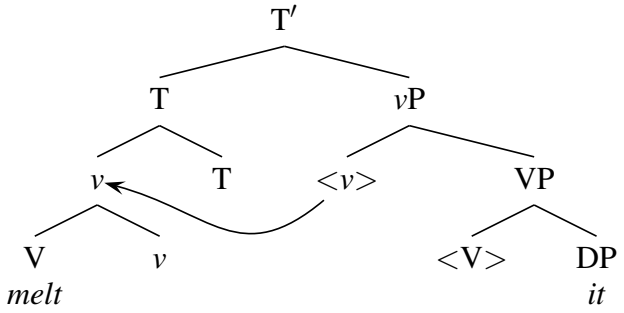
When a head moves up to another head, e.g., V moves to *v*, this doesn't change the fact that the higher head is still the lexical item whose features have projected in the constituent it heads. So, below, the features of *v* are still the features of *v*P, regardless of whether V has moved up to combine 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, some 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 still 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.

7 Agree, feature checking, and the Hierarchy of 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 is, a feature that is capable of checking 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."

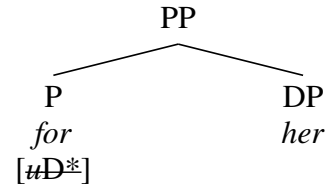
There are two different dimensions on which uninterpretable features can differ from one another. One dimension is the valued–unvalued dimension. An example of a valued uninterpretable feature is the [*u*D*] feature on P: this expresses a need for a very specific kind of matching feature, a category feature [D]. An example of an unvalued uninterpretable feature is the [*u*Infl:] feature of *v*: this expresses a need for some inflectional 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. For example, the [*u*D*] feature of T that makes the subject move into the specifier of TP will take the closest DP it finds (which will be the Agent, if there is one), and will not skip past the Agent to take a more distant Theme DP.

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.

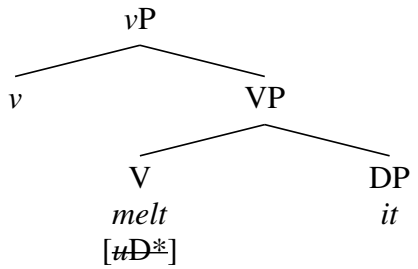
When the Hierarchy of Projections is involved, the Hierarchy of Projections "takes priority." It's not exactly clear why it does, but it nevertheless does. So, let's pause for a moment to review the Hierarchy of Projections.

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

There are a couple of different things that the HoP tells 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*, and V are not optional. We take this to mean that every V must have a *v* above it in the structure. Every clause must have a T.

The Hierarchy of Projections is a second way to motivate Merge (the first was 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.



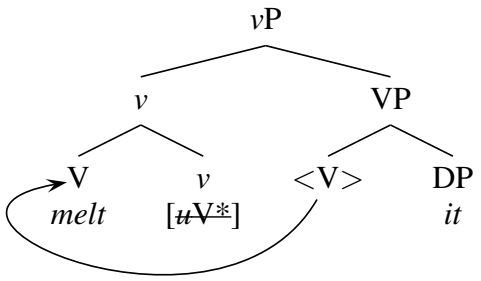
The motivation for Merging *v* with VP above is the Hierarchy of Projections. VP had no further uninterpretable category features to check and was therefore "finished."

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.

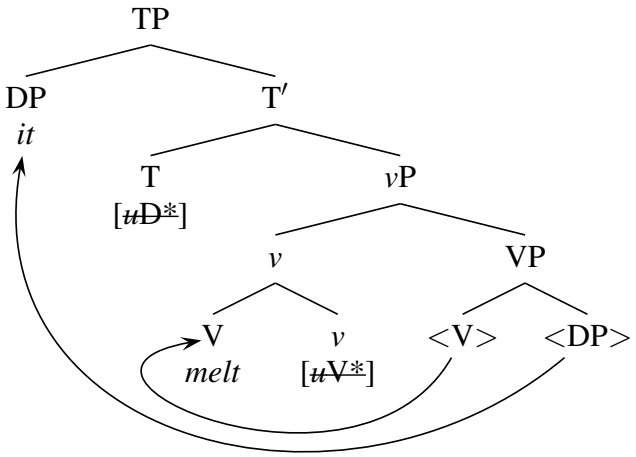
So: Even if there is another reason to Merge, the HoP takes priority, and when Merge happens due to the HoP, the two things Merged are not eligible to check each other's features. (Note, however, that features inside the two things Merged are still eligible for checking. So, T is Merged into the structure as a response to the HoP, but when that happens, it can then see *u*Infl:-features within the *v*P. It just wouldn't be possible for T to check a [*uv**] feature if it had one, without doing a further movement).

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, say, a past tense T is Merged with an auxiliary phrase like ProgP, the [tense:past] feature of T values the [*u*Infl:] feature of Prog as strong ([*u*Infl:past*]). The now-strong feature of Prog cannot be checked in place—not so much because T and ProgP 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 it had received a value from a higher auxiliary like M). So, in this case, the head moves up: Prog moves to (head-adjoin to) T.

8 Functional structure

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.
- 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.
- 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.

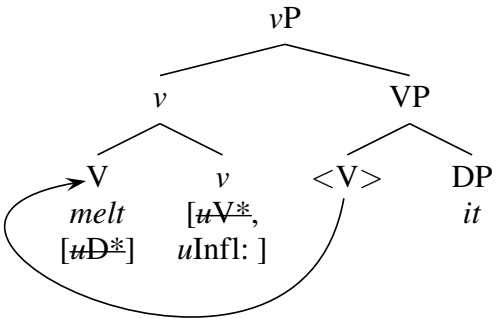
9 Inflection, and the auxiliary system

It is probably worth taking a special look at the auxiliary system in English and the way in which inflection is assigned (here we will not talk about how subject-verb agreement works, because that will be covered later in class).

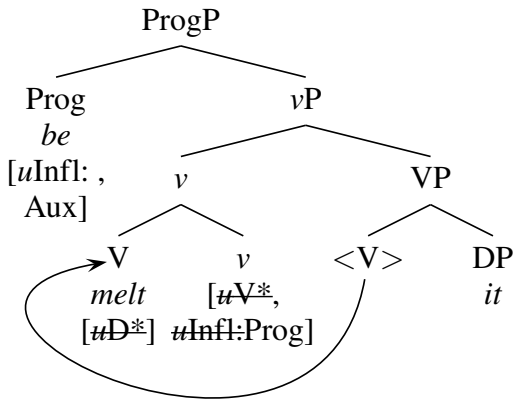
There is a set of things that take inflection, these are things that (at least abstractly) can be inflected. All of the auxiliaries (Perfective *have*, Progressive *be*, Passive *be*, Modals) and the verb (*v*) need to be inflected, which is reflected by having a [*u*Infl:] feature.

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.

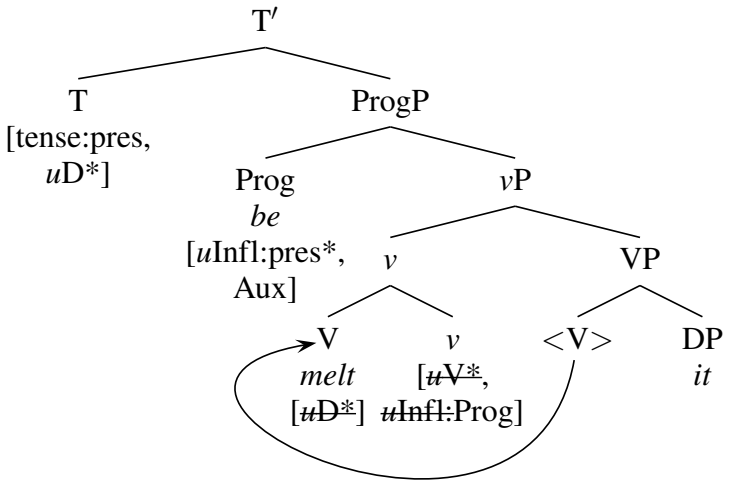
The things that are capable of assigning inflection are T, M, Perf, Prog, and Pass. T is a somewhat special case, so we'll come back to that. For the others, at the point they are Merged into the tree, they can value (and check) the closest [*u*Infl:] feature they c-command. For a simple example with 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 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 can value the [*u*Infl:] feature of Prog, which it does with its tense value. Furthermore, because Prog is an auxiliary (it has an [Aux] feature, as do Perf, Pass, and M), T values the [*u*Infl:] "as strong": [*u*Infl:pres*].



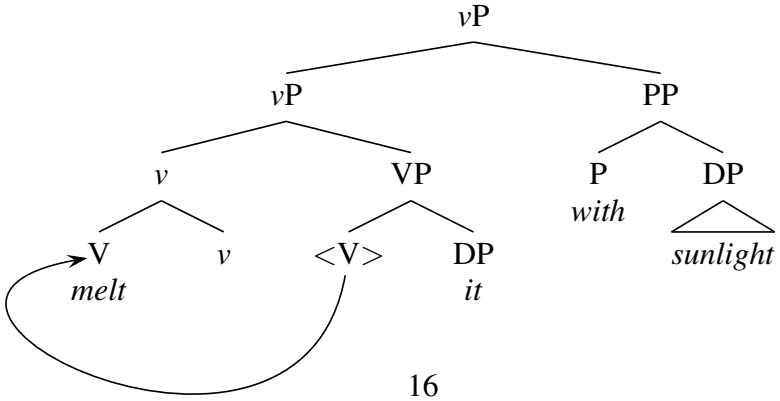
The next step will be to move Prog to T (by head-movement) in order to check the [*u*Infl:pres3sg*] feature of Prog (which, because it is strong, has to be checked very close to T—i.e. with the Prog head-adjoined to T). This happens in the same way that V moves to *v*, and then the derivation continues,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

10 Adverbs and adjuncts

Adverbs and adjectives are modifiers that attach to the structure via Adjoin. The intuition is that they modify the semantics but do not affect the syntactic status of the thing that they attach to—so if you Adjoin an adverb to a *v*P it was a *v*P before and it is still a *v*P afterwards. The semantics of adjunction is such that the thing being adjoined to is the thing being modified semantically. So, if you have an adverb that describes the manner of an event, it is it properly adjoined to the *v*P, and if you have an adverb that is more of a "scene-setter" describing time or location, it is properly adjoined higher, to the TP.

Adverbs generally adjoin to either *v*P or TP. Although it is true that adjectives cannot adjoin to *v*P (adjectives go with nouns) and adverbs cannot adjoin to a nominal phrase (adverbs go with verbs), this restriction is not encoded in our system anywhere. In particular, do not succumb to the intuitive thought that "an adverb needs a verb" and that there therefore must be a strong selectional feature on adverbs. Adjunction (of phrases) is an operation that does not check features, so we cannot say that adverbs have a feature to check. We must instead assume that what goes wrong if you mix adjectives and verbs or adverbs and nouns is something either in the semantics (it wouldn't be a semantically interpretable sentence if they were mixed like that) or in the pronunciation (if adverbs and adjectives are actually the same and only agreeing with what they attach to, such that being attached to a verb yields a pronounciation as an adverb and being attached to a noun yields pronunciation as an adjective).

Quite often, a PP will serve the same kind of function as either an adverb or an adjective, and it will be adjoined in the same way.



11 A list of heads and features

Just as a kind of quick reference, here's a list of heads and features that they would generally occur with. Features that sometimes occur but not in all situations are in parentheses.

Heads and features	
N	ϕ:*value*
n	*u*N* (*u*case:of)
Poss	*u*D*
D	*u*ϕ: , *u*case: (*u*case:gen*)
P	*u*D*, *u*case:acc
V	(*u*D*, *u*P*, *u*T*, *u*C*)
v	*u*V*, *u*Infl: (*u*case:acc, *u*D*)
Pass, Perf, Prog, M	Aux, *u*Infl:
Neg	
T	tense:*value*, *u*D*, *u*ϕ: , *u*clause-type: (*u*case:nom, *u*case:ing)
C	clause-type:*value* (*u*wh*, *u*top*, *u*case:null)