CAS LX 422 ∼ GRS LX 722 Intermediate Syntax
Lecture 5

Agree

1 Auxiliaries and Tense

Auxiliaries and modals and verbs

How do we determine which form each verb takes?

(1)	I ate	(5)	I had been eating
(2)	I could eat	(6)	I could have eaten
(3)	I had eaten	(7)	I could be eating
(4)	I was eating	(8)	I could have been eating

- HAVE (perfective aspect): I have eaten. I had eaten.
- BE (progressive aspect): I am eating. I was eating.
- COULD (modal): I can eat. I could eat. I shall eat. I should eat. I may eat. I might eat. I will eat. I would eat. I must eat.

Auxiliaries and modals and verbs

There seems to be an order: Modal, Perf, Prog, verb.

(9)	I could have been eating	M Perf Prog V
(10)	* I could be having eaten	M Prog Perf V
(11)	* I was canning have eaten	Prog M Perf V
(12)	* I had cannen be eating	Perf M Prog V
(13)	* I was having cannen eat	Prog Perf M V
(14)	* I had been canning eat	Perf Prog M V

Capture with an extended Hierarchy of Projections. Categories M, Perf, Prog. Each optional, all carrying the [aux] feature.

Hierarchy of Projections
(M) > (Perf) > (Prog) > *v* > V

Negation

(15)	I did not eat	**not** V
(16)	I could not eat	M **not** V
(17)	I had not eaten	Perf **not** V
(18)	I was not eating	Prog **not** V

(19) I had not been eating — Perf **not** Prog V

(20) I could not have been eating — M **not** Perf Prog V

Suppose *not* is of category Neg. How can we describe where *not* occurs? How can we fit it into our Hierarchy of Projections? Suppose that we can. Where?

Tense

Idea: the first auxiliary (whatever it is) appears before Not. Neg cannot fit within the auxiliaries. But maybe it's above all the auxiliaries, and then something else is above it. Like where *do* goes, only appearing in negative sentences.

(21) They did not eat. They ate.

(22) They do not eat. They eat.

All *do* seems to reflect there is tense, so let's suppose that this position above Neg is T. Further, let us suppose that T (and only T) has the interpreted feature [past] or [nonpast] (or [±past] etc.). Tense is **interpretable** on T, though it is also reflected on the first verb or auxiliary.

Moving the auxiliary

Hierarchy of Projections

T > (Neg) > (M) > (Perf) > (Prog) > *v* > V

Just as V moves to *v*, we will suppose that the (highest) auxiliary (including Perf, Prog, M) moves to T.

If Neg is there you can see it happen. But we assume it happens anyway even if you cannot see it.

(23) They T+shall not <shall> be giving a book to Pat.

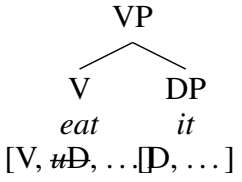
(24) They T+shall <shall> be giving a book to Pat.

If there is no auxiliary, the verb does not move (in English) to T; *not* somehow blocks tense association with the verb, requiring *do* to be inserted in the position of T.

They might eat it

they [D, . . .]	*v* [*v*, *u*D, . . .]
eat [V, *u*D, . . .]	*it* [D, . . .]
might [M, . . .]	T [T, past]

Pick up *eat* and *it* and Merge. Checks the *u*D feature of *eat*. The resulting object is now in the workspace, having inherited the features from its head.

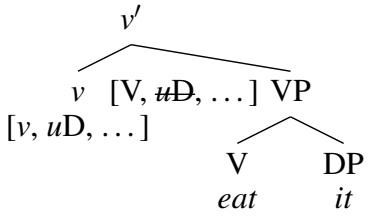


they [D, . . .]	*v* [*v*, *u*D, . . .]
VP [V, ~~*u*D~~, . . .]	
might [M, . . .]	T [T, past]

They might eat it

they [D, . . .]	*v* [*v*, *u*D, . . .]
VP [V, ~~*u*D~~, . . .]	
might [M, . . .]	T [T, past]

The Hierarchy of Projections says that once VP is complete, *v* is the next thing that needs to Merge. This counts as motivation. Labeled as *v*′ because it still has a [*u*D] and so will not be the maximal projection.

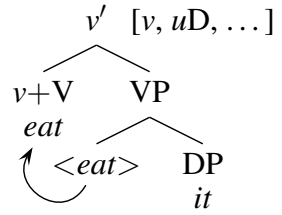


they [D, . . .]	
v′ [*v*, *u*D, . . .]	
might [M, . . .]	T [T, past]

They might eat it

Although we have not formalized exactly how this is implemented, something about *v* forces the V to move up to it.

This is presumed to happen as soon as *v* and V "can see" each other, once they have been put within the same syntactic object.

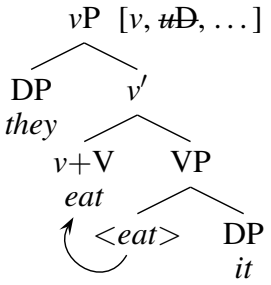


They might eat it

they [D, . . .]	
v′ [*v*, *u*D, . . .]	
might [M, . . .]	T [T, past]

To check the [*u*D] feature of *v*′, we Merge *they*.

*v*P [*v*, ~~*u*D~~, . . .]	
might [M, . . .]	T [T, past]

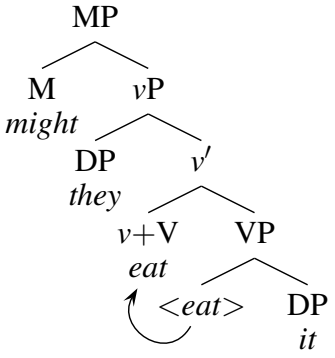


They might eat it

*v*P [*v*, ~~*u*D~~, . . .] *might* [M, . . .] T [T, past]

The Hierarchy of Projections says that when we reach the end of *v*P, we Merge the next thing we have, which is MP.

MP [M, . . .] T [T, past]

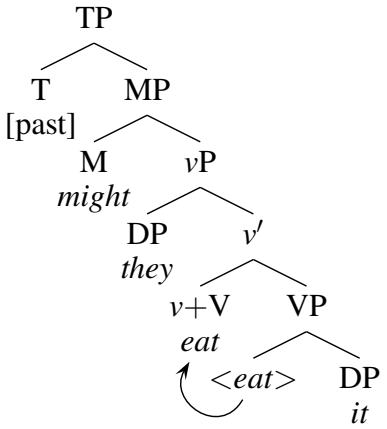


They might eat it

MP [M, . . .] T [T, past]

The Hierarchy of Projections says that when we are done with MP, we Merge T.

TP [T, past]

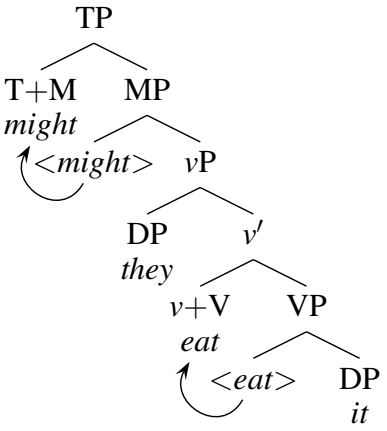


They might eat it

Then M moves up to T.

Why? Empirically, it seems that auxiliaries move to T, the modal seems like an auxiliary. Though the evidence is a bit sparse, it's more a conceptual hypothesis.

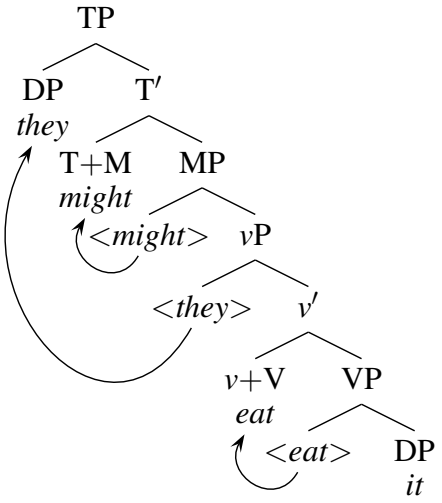
How? Well, we haven't worked out movement properly yet, so we don't have a way to say what happens or why. For the moment it is just a "rule from outside": the top auxiliary moves up to the position of T.



They might eat it

But now it is the wrong order, the subject should be before *might*. So, another movement based on a rule from outside: The T needs to have a DP in its specifier.

But there is nothing left in the workspace. So, we move the closest DP (*they*) up to satisfy this requirement.



They ate it

The T node:

- Provides a position above negation for auxiliaries
- Has a realization that seems to be tense-related

We hypothesize that this is the place where tense information "lives." This is the position in the tree where the lexical item that represents past or nonpast is placed.

Stated this way, it would seem to apply to any tensed sentence. The semantics of "past" are located here. Which then brings up the question of sentences like *They ate it*. Here, it seems that there is no independently realized tense, just a form of the verb. So where is T, given the assumption that it is where the tense features are introduced/interpreted?

2 Agree and Movement

A formal agreement

We will formalize the idea that tense features are semantically features of T, but are realized on the verb.

Informally, T has information that can determine the "ending" of the verb, and the verb "needs an ending." Needing something is modeled as an uninterpretable feature, something that cannot be allowed to survive to the point of interpretation at the interfaces.

We will ignore subject-verb agreement for the moment, and just focus on tense inflection. Subject-verb agreement will be similar but involves some other stuff we will talk about later.

Idea: Verbs that need inflection have a [*u*Infl: __] feature, a blank that needs to be filled in and can be filled in by tense features.

Needing an ending

For this idea to work, we need to suppose that this "I need [sufficient information to determine my] inflection" feature on the verb is somewhat selective about what constitutes sufficient/appropriate information.

It could get information about the tense (from T), that would be appropriate and sufficient. But verbs also can have *-ing* or *-en* endings, and those forms do not reflect tense, but rather some kind of aspect. So information like "perfective" or "progressive" is also appropriate and sufficient.

This is what "*u*Infl" is kind of standing in for. It says: "I need some inflectionally relevant features" and we (from outside) hypothesize that those include tense features like [past] as well as category features like T, Perf, Prog. And later we will include subject agreement features (person, number, gender) as well.

Feature classes

Specifically, this seems to need some kind of division of features into classes.

There are category features. N, V, T, C, Perf, Prog, etc. *Maybe* these can be modeled in terms of an array of binary values ([$\pm$N, $\pm$V, etc.]).

There are the features we'll use in subject agreement: person, number, gender. These tend to pattern together. So even if there isn't a "person" class of features, there is a "ϕ-feature" class of features that includes person, number, and gender features.

For tense, we suppose [past] is a tense feature. But what about nonpast? If features are binary, [$\pm$past] are both tense features. If not, we need to be able to indicate "tensed" separately from "past" (such that nonpast is tensed but not past).

Unvalued (uninterpretable) features

The kind of uninterpretable feature we are positing is slightly different from what we had before. So far, we've had a [*u*D] feature on V in order to force a Merge with an object. Such a Merge "checks" the feature by ensuring an exact match, V needs a D, and D is one.

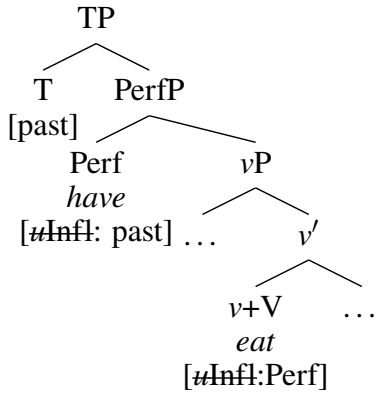
The [*u*Infl:] feature is a step more abstract, it says "I need something in this feature class." And, moreover, the thing it needs is somehow recorded on the verb after that. The verb needs these features, and then once it has them, those features are used in the determination of how it is pronounced.

Quite analogous to filling in a blank. In fact, the *u* designation might be redundant with the fact that the value is missing, if a missing value cannot be interpreted.

Had eaten

Goal: the perfective auxiliary *have* must get an ending and gets it from T. The verb must get an ending and gets it from Perf (resulting in *-en*).

This also models how the auxiliaries and tense each have the effect of inflecting the next verbal form down. There are set of things that need inflection (M, Perf, Prog, *v*) and a set of things that provide inflection (T, M, Perf, Prog).



Agree

Agree (take 1)
In the configuration X[F:value] ... Y[*u*F:], F checks and values *u*F, resulting in X[F:value] ... Y[~~*u*F~~:]

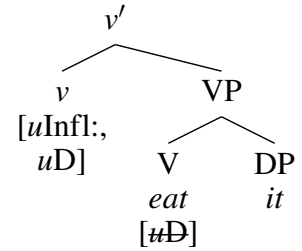
So we have two kinds of uninterpretable features so far, [*u*Infl:] type that need a value from a matching feature type in order to be checked, and [*u*D] type that need to be Merged with something containing a matching feature. These seem like they have slightly different priorities, insofar as the [*u*D] type seems to need to go first.

We also still need to clarify what conditions hold of "..." there as well. Is it $\emptyset$ (requiring sisters like [*u*D] does)? Or can there be more there inbetween so long as they're in the same syntactic object?

Inflecting the verb

We start off Merging *eat* with *it*, in order to check the [*u*D] feature of V. With no more uninterpretable features to check, we are done with VP.

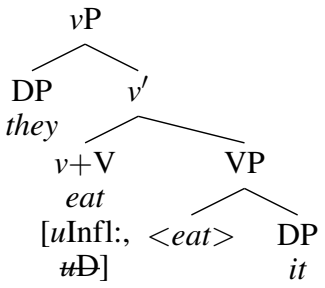
The Hierarchy of Projections dictates that Merging *v* is next. The need for an ending is encoded on *v* as a [*u*Infl:] feature. The need for a DP (will get the Agent θ-role) is encoded by the [*u*D].



Inflecting the verb

The first thing that happens when we merge *v* and VP is that V moves up to *v*. This is presumed to happen always at the first opportunity, at the first point they are both in the same syntactic object.

Then, the next most important thing to do is check the [*u*D] feature by Merging *they*. This is sufficient to consider the *v*P now "finished." (We could adjoin something at this point.)

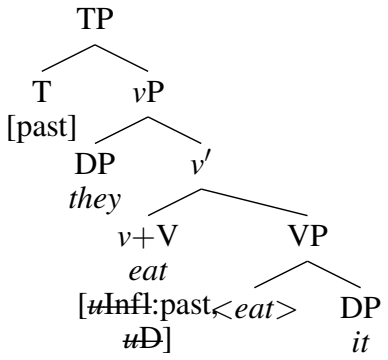


Inflecting the verb

If the next thing we have available (in the order dictated by the HoP) is T, we Merge it.

Once T is in the structure, it is able to value and check the [*u*Infl:] feature on *v*.

The verb, on the basis of having the [~~*u*Infl:~~past] feature, is pronounced as "ate."

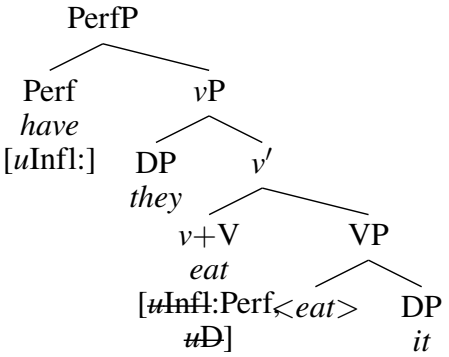


Inflecting the verb

If, on the other hand, the next thing we have available (in the order dictated by the HoP) is Perf, we Merge it (before we get to T).

Once Perf is in the structure, it is able to value and check the [*u*Infl:] feature on *v*.

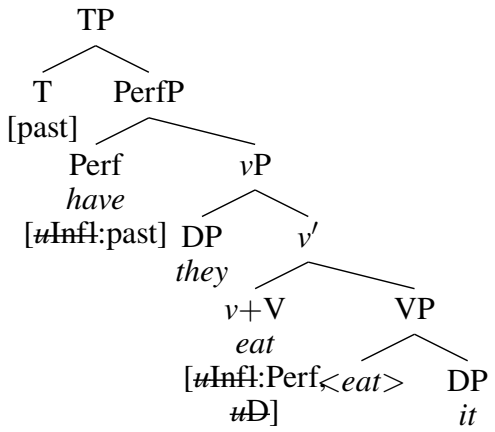
The verb, on the basis of having the [~~*u*Infl:~~Perf] feature, is pronounced as "eaten."



Inflecting the verb

We then proceed to T, Merging that next since that's the order the HoP dictates.

When T is Merged, it can value the [*u*Infl:] feature of Perf, resulting in [~~*u*Infl:~~past]. Thus Perf is pronounced as "had."



Movement

We've seen a few cases where things are moving around, and we've left them as mostly mystery magic.

Let's now formalize movement a bit more and explore what it does and why it happens in the context of the model being developed.

First assumption: The system is lazy, it will only do what it must. So, there must be a problem that movement solves which could not be solved without the movement. Movement must be forced by something.

The main kind of "problem" we have formalized so far is uninterpretability. Moving things must check a feature, defusing the threat of an uninterpretable feature.

Movement types

We've seen two types of movement so far.

XP-movement. Movement of the subject from the position where it gets a θ-role (e.g., Agent in Spec*v*P) to the "subject position," SpecTP.

Head-movement. Movement of a head to the position of another head. For example, movement of V to *v*, or movement of Perf to T.

The effect of movement would seem to be to change how close together things are. That is, we can think of the problem movement is solving as being one where the thing that moves was "too far away" until it moved.

Strong features vs. weak features

Suppose there are two types of uninterpretable feature, distinguished by whether they can be checked by something far away or by something that has to be close. We'll call the ones that require closeness "strong"—the intuitive idea is that they are kind of urgent, and also powerful enough to force something to move.

The "sentences need a subject" requirement, translated into "T needs a D in its specifier," can be implemented by saying that T has a strong (uninterpretable) feature that can only be checked by a D that is close by. We write such a feature like [*u*D*], meaning that it is a "strong uninterpretable D feature."

When we reach T in the tree, there are generally no more DPs around to Merge, so this feature gets checked by locating a DP we already have in the tree and moving it to the specifier of TP (that is, close).

Searching

Operationally, we Merge T with its [*u*D*] feature into the tree, and then look into the thing T Merged with to find a DP to move. The search goes deeper until it finds one and then the system moves that one. The system is lazy, so if it finds one, it does not keep looking, it just moves the first one it finds.

So T has a feature [*u*D*] which is strong. The Agent DP inside the *v*P has a [D] feature, which matches the strong feature (so could in principle check it). But because [*u*D*] is strong, it cannot be checked with the DP so far away. So we move the DP up to Merge with the object T heads, placing it in the specifier of the TP. Now the [*u*D*] feature and the D feature are sisters, so they're close enough, and the problem is solved.

Agree

Agree

If:

- X has feature [F1], Y has feature [F2]
- X c-commands Y or Y c-commands X
- [F1] and/or [F2] are/is uninterpretable.
- [F1] matches [F2]
- X and Y are close enough, meaning:
  - There is no closer matching feature between X and Y
  - If [F1] or [F2] is strong, X and Y share the same mother node

Then:

- Any unvalued feature ([F1] or [F2]) is valued.
- The uninterpretable feature(s) is/are checked.

Agreement as motivation

Agree thus constitutes a motivation for movement. It's not the operation that moves things (nor is it the operation that Merges things), but it is the reason those operations are undertaken. Agree is the way we defuse uninterpretable features.

Also: With this distinction between strong and weak uninterpretable features, where strong ones require sisterhood, we should recast the features that introduce arguments into θ-roles as strong as well. So, V has a [*u*D*] feature now. Forcing Merge of the Theme.

Matching and checking

Matching

- Identical features match. [D] matches [*u*D].
- Some features match several things. [*u*Infl:] can match values of the tense features ([past]) as well as category features ([Perf], [Prog]).
- If thre are two options, only the closest ones participate in Agree.

Checking

- An unvalued feature is always uninterpretable.
- Valuing a feature will check it.
- A privative feature is simply checked when it matches.

Other properties of Agree

Strong features Agree first

- Where a single head has more than one feature that must Agree, the strong ones are satisfied first

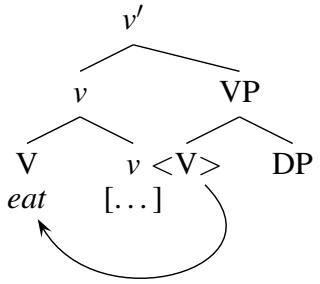
The system is lazy

- Agree always goes with the closes option it can find in order to check an uninterpretable feature.
- If Agree locates a matching feature on X for one uninterpretable feature, and X has a different feature that also matches, both features will be checked.
- Examples are coming up later, but for cross-referenceing: these properties are important for subject agreement.

Head movement

When V moves to *v*, we will assume that V **head-adjoins** to *v*. The *v* head is basically replaced by a *v* head that has a V hanging off of it.

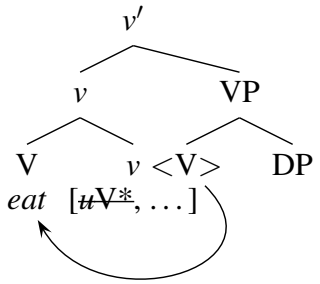
Adjunction does not change projection levels. *v* is still a minimal projection, and still the head of *v*P. But it is a **complex head**—it is a head with another head adjoined to it.



Head movement

We can implement this by saying that *v* has a [*u*V*] feature, forcing the V to be close.

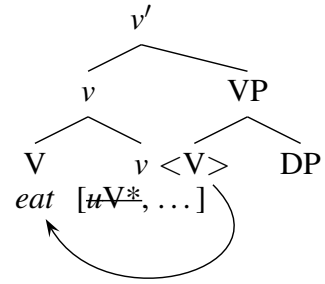
But wasn't it already close? Here, in order for this to make any sense, we need to say that you can't check a feature when you Merge for the HoP. We Merged *v* with VP for the HoP, so it neutralizes that relation for feature-checking. The V needs to get close to the *v* in some other way, and soon.



Copies and traces

In line with the "minimal machinery" aspiration, we will try to implement movement by using what we already have, Merge.

Movement will be selection of something inside a syntactic object and then Merging it or Adjoining it again. No "delete" step, just Merge it again (or, maybe, make a copy and Merge a copy).



Though this potentially increases need for pronunciation machinery, since you have several copies (maybe), and need to decide which to pronounce. Under most (all?) circumstances it is the highest copy, the one that c-commands the others.

Only auxiliaries move to T

(25) I do not eat green eggs and ham

(26) I was not eating green eggs and ham

(27) I have not been eating green eggs and ham

(28) I would not have been eating green eggs and ham

- There is a set of things that move to T: the auxiliaries (*have*, *be*, modals). Main verbs do not move to T. And only the *topmost* auxiliary moves to T.
- Since auxiliaries and main verbs behave differently, they must be differentiated. Suppose auxiliaries have the feature [aux] (i.e. the property of being auxiliaries).
- Movement is driven by strong features.
- So, [*u*Aux*] on T? No, that does not work.
- [*u*T*] on Aux? No, that would not be promising either.

Moving an auxiliary to T

Moving to T in English, observationally
A head with the feature [aux], and whose [*u*Infl:] feature is valued by T, moves to adjoin to T.

Movement is driven by strong features. None of these features can be strong in the general case. So, it appears that we need to stipulate a situation in which a feature *becomes* strong based on its context/situation. Here is specifically what we will say:

Moving to T in English
A head with the feature [aux], and whose [*u*Infl:] feature is valued by T **is valued as strong** (therefore not checked unless it is close). Will need to move to be close in order for the feature to be checked.

Where we are

Merge
Take two syntactic objects and form a single syntactic object out of them. The features of the resulting object are those of the one that projected.

Agree
Where either/both [F1] and [F2] are uninterpretable, they can see each other, and match (wihtout a closer match), the uninterpretable feature(s) is/are valued. If no features are strong, the now-valued uninterpretable features are checked. Otherwise, features are checked only if the matching features are close together.

Move
Select some Y within a syntactic object X and Merge Y again to X, or (if Y is a head) adjoin it to the head of X.