CAS LX 522
Syntax I

10

TP, Agree, and our quickly growing tree
(5.1-5.3)

Auxiliaries and modals and verbs

- I ate.
- I could eat.
- I had eaten.
- I was eating.
- I had been eating.
- I could have eaten.
- I could be eating.
- I could have been eating.

- So: *could*, *have*, *be*, *eat*. How do we determine what form each verb takes?

Auxiliaries and modals and verbs

- *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.

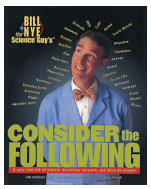
Auxiliaries and modals and verbs

- I could have been eating.
- *I could be having eaten.
- *I was canning have eaten.
- *I had cannen be eating.
- *I was having cannen eat.
- *I had been canning eat.
- It looks like there's an order:
  - Modal, Perf, Prog, verb.

Auxiliaries and modals and verbs

- Suppose:
  - *Have* is of category Perf.
  - *Be* is of category Prog.
  - *May*, *might*, *can*, *could* are of category M.
- They are heads from the lexicon, we will Merge them into the tree above *v*P. Their order is captured by a new extended Hierarchy of Projections:
  - Modal > Perf > Prog > *v* > V
- Except not every sentence has these. So:
  - (Modal) > (Perf) > (Prog) > *v* > V

Negation



- Consider the following:
  - I did not eat.
  - I could not eat.
  - I had not eaten.
  - I was not eating.
  - I had not been eating.
  - I could not have been eating.
- Suppose *not* is of category Neg.
- How do we describe where *not* occurs? How can we fit it into our Hierarchy of Projections?

Where does Neg fit?

- Suppose that we *can* fit Neg in our Hierarchy of Projections. Just like the other things we just added.
  - (Modal) > (Perf) > (Prog) > v > V
- Where would it go in the HoP, and how can we explain the word order patterns?
  - I could not have been eating.
  - I had not been eating.
  - I was not eating.
  - I did not eat.
- Remember *v* and how we explained where the verb is in *Pat gave a book to Chris*?

A-ha.

- Picture this:
  - I ?+might not <might> have been eating.
  - I ?+had not <had> been eating.
  - I ?+was not <was> eating.
- So what is ?, then?
  - He did not eat. He ate.
  - He does not eat. He eats.
- All that *do* seems to be doing there is providing an indication of…tense.

HoP revisited

- So, now we know where Neg goes. Above all the other things, but below tense (category T).
  - T > (Neg) > (M) > (Perf) > (Prog) > *v* > V
- Just as V moves to *v*, so do Perf, Prog, and M move to T.
- If Neg is there, you can see it happen.
  - They T+shall not <shall> be eating lunch.
  - They T+shall <shall> be eating lunch.

What does <u>do</u> do?

- But what about when there's just a verb and Neg, but no M, Perf, or Prog?
  - I ate lunch.
  - I did not eat lunch.
- *Eat* clearly does not move to T.
- But *not* "gets in the way", so tense cannot "see" the verb. Instead, the meaningless verb *do* is pronounced, to "support" tense. "*Do*-support"
  - We will return to the details in due course…

So, we have T

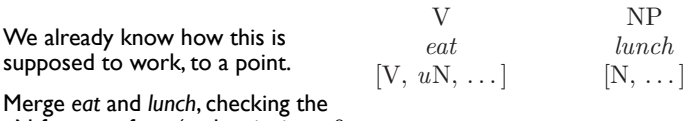
- We've just added a category T, tense.
- **The idea:** The tense of a clause (past, present) is the information that T brings to the structure.
- T has features like [T, past] or [T, pres]
  - Or perhaps [T, past] or [T, nonpast].
- These features are *interpretable* on T. T is where tense "lives." We see reflections of these tense features on verbs (*give*, gave, *go*, *went*) but they are just reflections. Agreement. The interpretable tense features don't live on verbs, they live on T.

Pat might eat lunch.

Pat [N, …] *v* [*v*, *u*N, …]
eat [V, *u*N, …] *lunch* [N, …]
workbench
might [M, …] T [T, past]

We already know how this is supposed to work, to a point.

Merge *eat* and *lunch*, checking the *u*N feature of *eat* (and assigning a θ-role to *lunch*, namely Theme—this is NP daughter of VP).



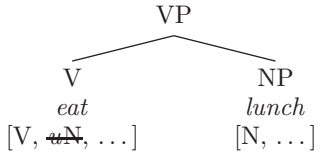
Pat might eat lunch.

Pat [N, …] *v* [*v*, *u*N, …]

workbench

might [M, …] T [T, past]

VP [V, ...]



We already know how this is supposed to work, to a point.

Merge *eat* and *lunch*, checking the *u*N feature of *eat* (and assigning a θ-role to *lunch*, namely Theme—this is NP daughter of VP).

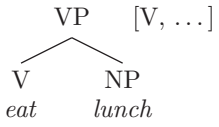
Pat might eat lunch.

Pat [N, …] *v* [*v*, *u*N, …]

VP [V, ...]

workbench

might [M, …] T [T, past]



We already know how this is supposed to work, to a point.

Merge *eat* and *lunch*, checking the *u*N feature of *eat* (and assigning a θ-role to *lunch*, namely Theme—this is NP daughter of VP).

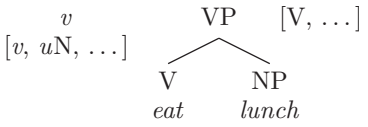
Pat might eat lunch.

Pat [N, …] *v* [*v*, *u*N, …]

VP [V, ...]

workbench

might [M, …] T [T, past]



Merge *v* and the VP *eat lunch*, in conformance with the Hierarchy of Projections. *v* projects, and still has a *u*N feature.

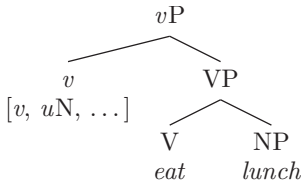
Pat might eat lunch.

Pat [N, …]

workbench

might [M, …] T [T, past]

*v*P [*v*, *u*N, …]



Merge *v* and the VP *eat lunch*, in conformance with the Hierarchy of Projections. *v* projects, and still has a *u*N feature.

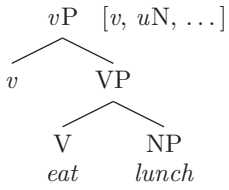
Pat might eat lunch.

Pat [N, …]

*v*P [*v*, *u*N, …]

workbench

might [M, …] T [T, past]



Merge *v* and the VP *eat lunch*, in conformance with the Hierarchy of Projections. *v* projects, and still has a *u*N feature.

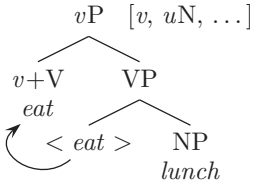
Pat might eat lunch.

Pat [N, …]

*v*P [*v*, *u*N, …]

workbench

might [M, …] T [T, past]



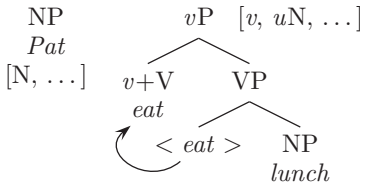
Move the V *eat* up to *v*.

Pat might eat lunch.

Pat [N, …]
*v*P [*v*, *u*N, …]

workbench
might [M, …] T [T, past]

Merge *Pat* with *v′* to check the *u*N feature and assign a θ-role (Agent, this is NP daughter of *v*P).

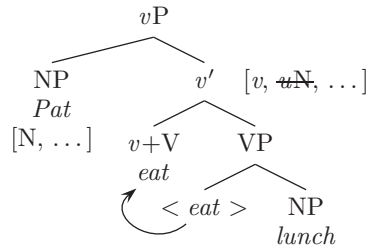


Pat might eat lunch.

workbench
might [M, …] T [T, past]

*v*P [*v*, …]

Merge *Pat* with *v′* to check the *u*N feature and assign a θ-role (Agent, this is NP daughter of *v*P).



Pat might eat lunch.

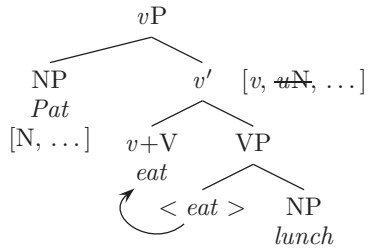
*v*P [*v*, …]

workbench
might [M, …] T [T, past]

So, now what do we do with *might*?

1) And eat lunch Pat shall.
2) What Pat should do is eat lunch.

It kind of seems like it goes between the subject and the verb, but how?



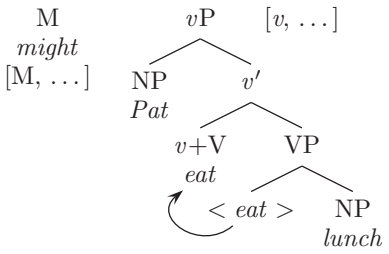
Pat might eat lunch.

*v*P [*v*, …]

workbench
might [M, …] T [T, past]

If we leave everything as it is so far (UTAH, Hierarchy of Projections), the only option is to Merge *might* with the *v*P we just built.

So, let's.

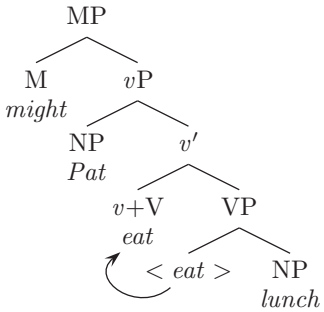


Pat might eat lunch.

workbench
T [T, past]

MP [M, …]

- Now, we have one more thing on our workbench (T) and the HoP says that once we finish with M, we Merge it with T.
- And so Merge T, we shall.

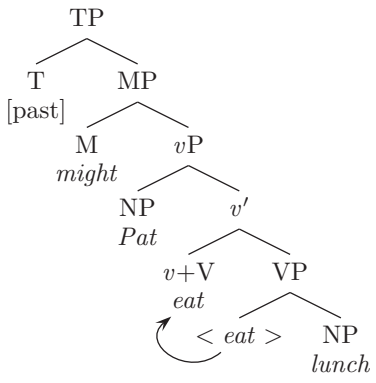


Pat might eat lunch.

Then, M moves up to T.

Why? Because M, Perf, and Prog all move up to T. For the same kind of reason that V moves up to *v*.

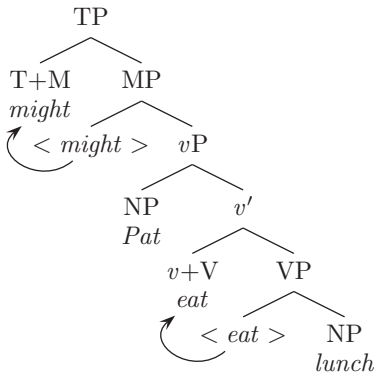
Right now we have no way to describe this in our system, except with this "rule from the outside" that stipulates that V moves to *v*, and {M/Perf/Prog} moves to T.



Pat might eat lunch.

Ok, that's all fine and good, except that the sentence is *Pat might eat lunch* not *Might Pat eat lunch*

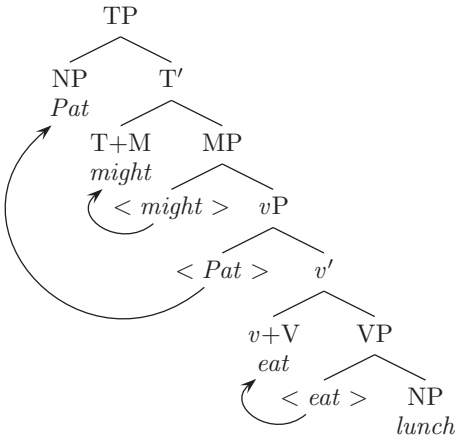
How do we get *Pat might eat lunch* out of this?



Pat might eat lunch.

As previewed last time, the subject *moves* to this first position in the sentence, around the modal.

"Moving" *Pat* here means Merging a copy…

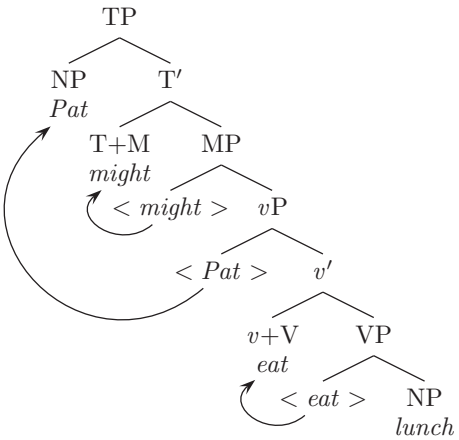


Pat might eat lunch.

Great. Why?

Jumping ahead, we're going to say that this is a property of T-type things generally: T needs to have an NP in its specifier.

We can encode this as a (special type of) uninterpretable feature on T: [*u*N*]. More on that later.



☞⚠ WARNING ⚠☜

- What we've done here is **not** quite the same as what is in the textbook. (But it's better).
  - In the textbook, modals are not treated as their own category, but rather as a kind of T.
  - The revision we made here will pay off soon. Keep this difference in mind as you review the textbook on this point. You will see no MPs in the book. But you should see them on the homeworks/tests you turn in.

Side note: "I" vs. "T"

- You may have heard in the past that it tense should be of category I (for **I**nflection), rather than T (For **T**ense).
- **Rest easy:** T and I are (for current purposes) just two names for the same thing.
  - Historically, this was called INFL, then I, and now usually called T. But these are just names. I vs. T, Istanbul vs. Constantinople; St. Petersburg vs. Leningrad.

Pat ate lunch

- Now that we have T in the Hierarchy of Projections, we're stuck with it.
- Yet, where is T in *Pat ate lunch* or *Pat eats lunch*?
- It looks like the tense marking is on the verb, we don't see anything between the subject and the verb where T ought to be.
- Now that we have T, this is where tense features *belong*. We take this to be the thing that determines the tense of the sentence, even if we sometimes *see* the marking on the verb.

Pat ate lunch

- Since (most) verbs sound different when in the past and in the present tense, we suppose that there is a [past] or [present] feature on the verb.
- However, to reiterate: **tense belongs on T.**
- The tense features on the verbs are uninterpretable.

Feature classes

- There are **tense** features. Like past, like present. There are **case** features. Like nom, like acc. There are **person** features. Like 1st, like 2nd. There are **gender** features. Like masculine, like feminine.
- So, we can think of this as a feature category or feature **type** that has a **value**.

[Gender: masculine]	[Person: 1st]
[Tense: past]	[Case: nom]

Agree

- T nodes have features of the tense type. Maybe past, maybe present.
- Suppose that *v* has an uninterpretable feature of the tense type, but *unvalued*.
  - What we're trying to model here is *agreement*.

Agree
In the configuration X[F: val] … Y[*u*F:]
F **checks** and **values** *u*F, resulting in
X[F: val] … Y[~~*u*~~F: val].

Unvalued features

- The idea is that a lexical item might have an *unvalued* feature, which is uninterpretable as it stands and needs to be given a *value* in order to be interpretable.
  - The statement of Agree on the previous slide is essentially saying just that, formally.
- This gives us **two kinds of uninterpretable features** (unvalued and regular-old uninterpretable features), and two ways to check them (valuing for unvalued features, checking under sisterhood for the other kind).
  - Unvalued [*u*F:]. Regular-old [*u*F].

To be continued...

(Actually, it's unlikely we'll get to here anyway, so I suppose it will be continued immediately after you actually see the previous slide, next time.)