

# CAS LX 522

## Syntax I

Week 10a.  
CP & PRO  
(8.1-8.2.5)

## Types of sentences

- Sentences come in several **types**. We've mainly seen **declarative clauses**.
  - 1) Horton heard a Who.
- But there are also questions (**interrogative clauses**)...
  - 2) Did Horton hear a Who?
  - 3) Who did Horton hear?
- ...**exclamatives**...
  - 4) What a crazy elephant!
- ...**imperatives**...
  - 5) Pass me the salt.

## Declaratives & interrogatives

- Our syntactic theory should allow us to distinguish between clause types.
- The basic content of *Phil will bake a cake* and *Will Phil bake a cake?* is the same.
- Two DPs (*Phil*, nominative, and *a cake*, accusative), a modal (*will*), a transitive verb (*bake*) that assigns an Agent  $\theta$ -role and a Theme  $\theta$ -role. They are minimally different: **one's an interrogative, and one's a declarative**. One asserts that something is true, one requests a response about whether it is true.

## Clause type

- Given this motivation, we seem to need one more category of lexical items, the **clause type** category.
- We'll call this category **C**, which traditionally stands for **complementizer**.
- The hypothesis is that a declarative sentence has a **declarative C** in its structure, while an interrogative sentence (a question) has an **interrogative C**.

## Embedding clauses

- The reason for calling this element a **complementizer** stems from viewing the problem from a different starting point.
- It is possible to **embed** a sentence within another sentence:
  - 1) I heard [Lenny retired].
- And when you embed a declarative, you generally have the option of using the word *that*.
  - 2) I heard **that** [Lenny retired].
- So what is that *that*?

## What's that ?

- We can show that *that* "belongs" to the embedded sentence with constituency tests.
  - 1) What I heard is that Lenny retired.
  - 2) \*What I heard that is Lenny retired.
- There's a demonstrative *that*, but that's not what *that* is.
  - 3) \*I heard this Lenny retired.
- So, *that* is its own kind of thing. It's an introducer of embedded clauses, a **complementizer**.

## Complementizers

- There are a couple of different kinds of complementizer. *That* is for embedding declarative sentences.
  - 1) I understand **that** Alton dislikes unitaskers.
- It's also possible to embed an interrogative sentence, like so:
  - 2) I wonder **if** Alton dislikes unitaskers.
  - 3) I wonder **whether** Alton dislikes unitaskers.
- Here, *if* and *whether* serve as complementizers, introducing the embedded interrogative.
  - I wonder about the answer to *Does Alton dislike unitaskers?*

## Selection

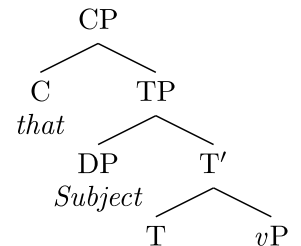
- Just like the verb *dislikes* takes the DP *unitaskers* as its object, some verbs take clauses as their object.
- Some verbs specify what kind of clause they take:
  - 1) I claimed that Alton dislikes unitaskers.
  - 2) \*I claimed if Alton dislikes unitaskers.
  - 3) \*I wondered that Alton dislikes unitaskers.
  - 4) I wondered if Alton dislikes unitaskers.
- This is a matter of **selection**. Some verbs select for declaratives, some verbs select for interrogatives. Some verbs can take either, some neither.
  - 5) I know that Alton dislikes unitaskers.
  - 6) I know if Alton dislikes unitaskers.
  - 7) \*I washed that Alton dislikes unitaskers.
  - 8) \*I washed if Alton dislikes unitaskers.

## C

- So, we have lexical items like *that* and *if*, which are complementizers (category: C), and have a value for clause type.
 

<i>that</i>	[C, clause-type:decl, ...]
<i>if</i>	[C, clause-type:Q, ...]
- Where is it structurally? We know it forms a constituent with the clause it introduces. We know that verbs can select for different kinds of C. The natural conclusion is that it is a sister to TP, at the top of the tree, which projects.

## CP



- C is the head of CP.
- Saying this also provides a natural explanation of why in SOV languages, complementizers are generally on the right.
  - 1) Hanako-ga [Taroo-ga naita to] itta.  
H.- nom T. -nom cried that said  
'Hanako said that Taro cried.'

## that or not that

- C specifies the clause type; *that* indicates a declarative clause. Why then are both of these good?
  - 1) Jack claimed that Jill fell.
  - 2) Jack claimed Jill fell.
    - In French, Spanish, probably most other languages you don't have the option to leave out the C.
  - 3) J'ai dit **qu'** elle était malade  
I've said **that** she was ill  
'I said that she was ill'
  - 4) \*J'ai dit elle était malade
    - *Claim* doesn't embed interrogatives.
  - 5) \*Jack claimed if Jill fell.
    - So *Jill fell* is declarative in *Jack claimed Jill fell*.

## ∅

- Where does that leave us?
  - 1) Jack claimed Jill fell
- *Claim* only takes declarative complements.
- *Jill fell* is declarative.
- Clause type is a feature of C.
- Thus: There is a declarative C. You just can't hear it.
- English has two declarative complementizers. One is *that*, one is ∅. In most cases, either one works equally well.

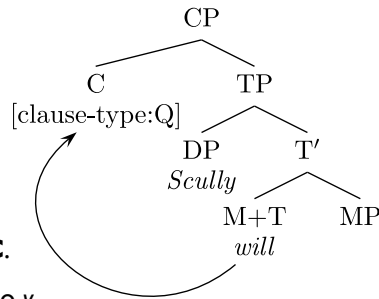
# Jill fell is a declarative

- But hold on a minute. *Jill fell*, just as its own sentence (not embedded) is also declarative.
  - Cf. *Did Jill fall?*
- So, we'll suppose that since the function of C is to mark clause type, there's a C in simple sentences as well.
- The C that heads the whole structure has somewhat special properties. Declarative C in that position is never pronounced. Interrogative C is not pronounced as a word, but makes its presence known by causing movement.

# SAI in YNQs

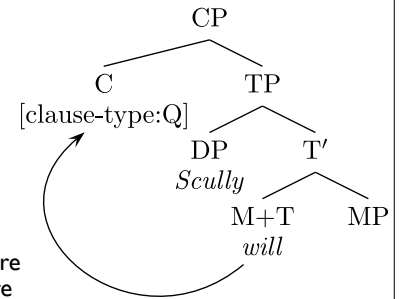
- In yes-no questions, the subject and auxiliary "invert" (Subject-Auxiliary Inversion):
  - 1) Scully will perform the autopsy.
  - 2) Will Scully perform the autopsy?
- Assuming everything we've got so far:
  - T has a [ $uD^*$ ] (EPP) feature to check, so *Scully* is in SpecTP.
  - The question is an interrogative.
  - (Unpronounced) C is to the left of TP.
- So what must be happening in yes-no questions?

# T-to-C



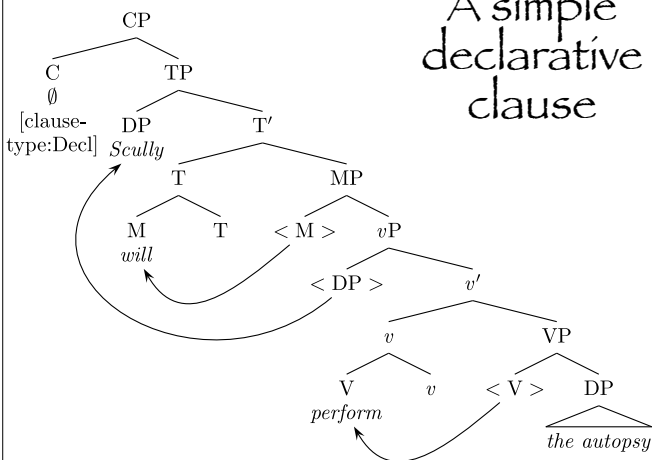
- A natural way to look at this:  
**T is moving to C.**
- Just like V moves to *v*, or like Aux (Perf, Prog, or Pass) moves to T, or like N moves to *n*.
- In (main clause) questions, T moves to C.

# T-to-C



- Specifically:
- Suppose T has an uninterpretable feature that matches a feature of C: [ $u_{\text{clause-type}}$ ].
- Suppose that when C values [ $u_{\text{clause-type}}$ ] as Q, the uninterpretable feature is strong.
- Cf. When T values [ $u_{\text{Infl}}$ ] on Aux (Prog, Perf, Pass), the feature is strong, and Aux moves to T.

# A simple declarative clause

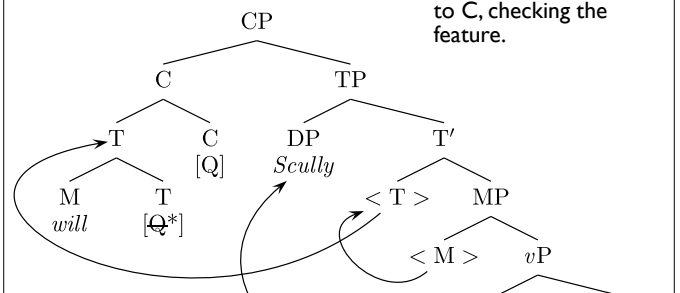


# YNQ

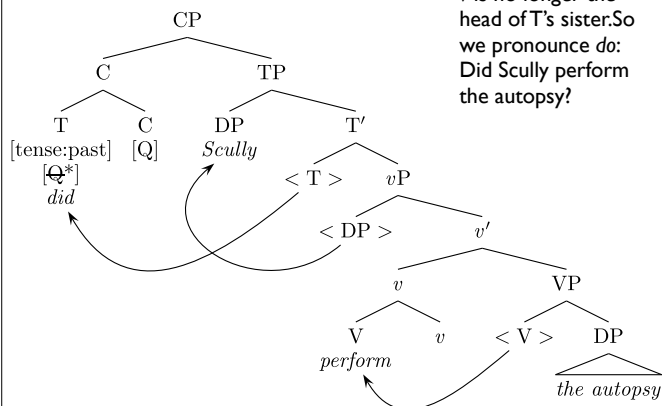
Abbreviations:
 

- [ $Q$ ] = [ $\text{clause-type:}Q$ ]
- [ $Q^*$ ] = [ $u_{\text{clause-type:}Q^*}$ ]
- [ $u_{\text{clause-type}}$ ] = [ $u_{\text{clause-type}}$ ]

- In a YNQ, the [ $Q$ ] feature of C matches and values the [ $u_{\text{clause-type}}$ ] feature of T as strong ( $[Q^*]$ ).
- T moves up to adjoin to C, checking the feature.



## YNQ



- If T is just a past or present tense marker, v is no longer the head of T's sister. So we pronounce *do*: Did Scully perform the autopsy?

## Embedding questions

- So, you can embed declaratives and you can embed questions
  - 1) I heard (that) Jill fell.
  - 2) I asked if Jill fell.
- Notice that the main clause is different:
  - If the topmost C is interrogative, we get SAI. If the topmost C is declarative, it is pronounced  $\emptyset$ .
  - If an embedded C is declarative, it can be pronounced either as  $\emptyset$  or as *that*. If an embedded C is interrogative, C is audible (*if*) and no SAI.
- So, T moves to C only in main clause interrogatives. [*clause-type*:] is strong only when valued as Q by a main clause C.

## Nonfinite clauses

- Some verbs embed finite declaratives, as we have seen: *I heard (that) Jill fell.*
- There are other verbs that embed **nonfinite** clauses. These come in a few types, but we'll start with the *try* type.
  - 1) Scully tried to perform the autopsy.
- This is two clauses: Scully tried something, and what it was was *to perform the autopsy*.

## $\theta$ -roles

- 1) Scully performed the autopsy.
  - 2) Scully tried to perform the autopsy.
- The verb *perform* has an Agent and a Theme, here *Scully* and *the autopsy*, respectively.
  - The verb *try* also has two  $\theta$ -roles, an Agent (the one trying) and a Theme (the thing attempted). Suppose that the Theme of *try* is [*to perform the autopsy*] here.

## $\theta$ -roles

- 1) Scully performed the autopsy.
  - 2) Scully tried to perform the autopsy.
- In the second sentence, *Scully* is both the one trying and, if you think about it, the one performing the autopsy. The same individual is the Agent of both.
  - Agent  $\theta$ -roles are assigned to the DP that is Merged into SpecvP.
  - **However:** You are not allowed to assign two different  $\theta$ -roles to the same DP. Otherwise, it should be possible for *Scully admires* to mean *Scully admires herself*.

## PRO

- 1) Scully tried to perform the autopsy.
- So, we have something of a problem here. We need an Agent DP in the vP for *perform*, and an Agent DP in the vP for *try*. But it appears as if there is only one DP around, *Scully*.
  - What to do? Once again gritting our teeth, we resolve ourselves to the fact that we need two DPs and can only see one— therefore, there must be a DP we can't see.
  - The DP we can't see, we call **PRO**.

# Control

1) Scully tried [PRO to perform the autopsy].

- PRO is a DP that is the Agent of *perform*, *Scully* is a DP that is the Agent of *try*.
  - It is impossible to actually *pronounce* an Agent for *perform*.
- 2) \*Scully tried [Mulder to perform the autopsy].
- The PRO Agent of *perform* must be interpreted as being the same person as the Agent of *try*.
  - PRO is a little bit like an anaphor in this respect; this fact is similar to the fact that *herself* in *Scully admires herself* must refer to *Scully*.
  - This obligatory co-reference goes by the name **control**. *Scully controls* PRO. Sentences with PRO in them are often called **control clauses**.

# PRO

- So why is it impossible to say this?
  - \*Scully tried [Mulder to perform the autopsy].
- The answer we'll give is that **nonfinite T (to) does not have a case feature**.
- Finite T has a [nom] feature which matches, values, and checks the [case] feature of the subject, checking itself in the process.
- Nonfinite T has no case feature at all, so *Mulder* would be left with its case unchecked.

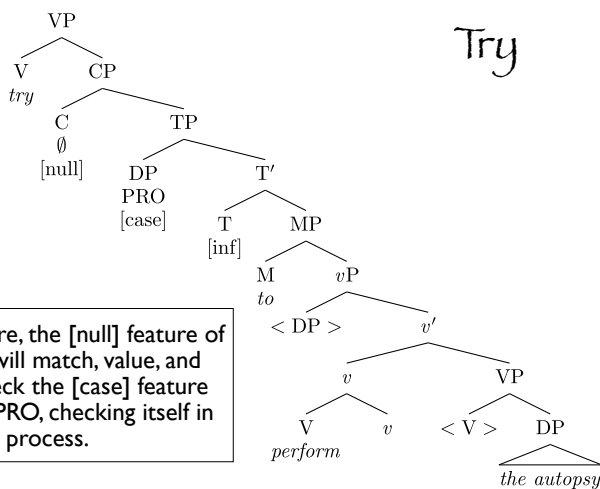
# Null case

- As for PRO, it is a DP so it has a [case] feature. If *Mulder* can't get its case checked by the nonfinite T, how does PRO get its case checked?
- A standard (and perhaps less than completely elegant) way to look at this:
  - **PRO is special**, it can only "show up" with "null case" ([ $\emptyset$ case:null]).
  - **Null case is special**, it is only allowed on PRO.
  - **Control clauses are special**, they are introduced by a null C that has a [null] case feature, which can check the [case] feature on PRO.

# Try

- So, *try* embeds a nonfinite CP, headed by the special null C with the [null] case feature.
- In turn, the subject must be PRO, in order to successfully check that feature of C.
- If the [case] feature of any other DP is valued and checked as [null], the derivation crashes: only PRO can have null case.
- The embedded clause must be nonfinite (T can't itself have a [nom] feature).
- If the [nom] feature of T checks the [case] feature of the subject, nothing is left to check C's [null] feature.

# Try



Here, the [null] feature of C will match, value, and check the [case] feature of PRO, checking itself in the process.

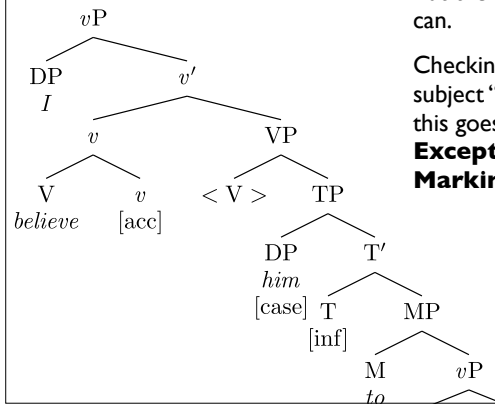
# Believe

- Another place where nonfinite clauses can be embedded is under the verb *believe*.
  - 1) I believe [him to be innocent].
- Here, we have an accusative subject, and a nonfinite T that is not capable of checking case.
- How is the (accusative) case of *him* checked?
- This relates to the fact that *believe* can also simply take a DP object:
  - 2) I believe him.
- So, how is the accusative case of *him* checked here?

## ECM

- The idea is that *believe* (actually the *v* that combines with the V *believe*) has an [acc] feature that can check the case of *him* in *I believe him*.
- Suppose that *believe* can either have a DP or a TP as its complement.
- What do we expect?

## ECM



Nonfinite T cannot check the case feature of *him*. But the higher *v* of *believe* can.

Checking the case of a subject “from above” like this goes by the name **Exceptional Case Marking (ECM)**.

## Arranging to leave

- A somewhat similar phenomenon occurs with verbs like *arrange*.
  - 1) Harry arranged for Tom to leave MI-5.
- Here, we have:
  - Nonfinite T, which cannot check case.
  - An overt subject (*Tom*) in the accusative.
  - The word *for*, which we classify as C.
- *For*, as a P, checks accusative case (*He baked a cake for her*). If the C *for* also has an [acc] feature, it could check the [case] feature on *Tom*.

## Arranging to leave

- Arrange-type verbs can take a CP complement.
  - 1) Harry arranged for Tom to leave MI-5.
  - 2) Tom arranged PRO to leave MI-5.
- But this is expected.
  - Nonfinite T, cannot check case.
  - The null C with [null] case can check the case of PRO.
  - An overt subject can't get null case:
    - \*Harry arranged Tom to leave MI-5.
  - PRO cannot get anything but null case:
    - \*Tom arranged for to leave MI-5.

## Summary

- Complementizers indicate clause type (*that/∅* for declaratives, *if/whether* for interrogatives).
- Some verbs embed clauses. Finite clauses are always CPs.
- Some verbs can embed nonfinite clauses, some embedding TP and others embedding CP.
  - *Believe* (*expect*, ...) embed TP and check accusative case (ECM verbs).
  - *Try* (*want*, ...) embed CP. This can either be:
    - C[null], checking null case on PRO.
    - *for*[acc], checking acc case on an overt subject. Not all verbs allow this option (*want* does, *try* doesn't).