Building the clause

Introduction to Syntax, Lecture 8-9

Sandhya Sundaresan, EGG Summer School, Wrocław 2019

August 8-9, 2019

A new class of puzzles

In this handout we're going to go after a group of new problems, which at first glance look unrelated, but turn out to have a crucial feature in common.

- ☞ They all involve, in one way or another, something showing up somewhere other than where we might have expected it.
- ☞ We'll see that phenomena of this type are widespread and seem to constitute one of the defining properties of natural language.

We can distinguish two broad sub-types:

1. A syntactic object appears in a different position in the sentence than we would have expected based on things like selection and the HoP.
2. The form of one syntactic object depends on or reflects properties of a distinct syntactic object in a different position.

Here's a quick list of relevant phenomena, some of which we'll get to this week, but most of which we won't:

- Agreement
- Binding
- Negative Polarity Items
- Surface subject position
- Case
- Passivization
- Subject-verb inversion
- Sequence of tense
- Question formation
- Determination of non-finite verb forms

Introducing Agree

An issue with tense-marking

Last time we introduced T as a functional head above *v*P to host modals, infinitival to and tense marking. But there's an issue when there's no auxiliary:

(1) Ellie misses Anna.

(2) Ellie missed Anna.

- ☞ Here tense is marked by a suffix on the main verb.
- ☞ But main verbs are of category V (perhaps also including a *v*), not T.
- ☞ So how do we get properties of T to affect the pronunciation of V?

Two basic possibilities:

1. T and V (and perhaps *v*) somehow get combined together when no auxiliary is present, so that T is pronounced as a suffix on V.
2. There is an operation affecting some features on V or *v* in a way related to features in T.

Either one of these requires a serious addition to our theory.

☞ For the present, we'll pursue possibility 2, seeing along the way that, while we will need something like possibility 1 later, it isn't quite suited for this particular problem.

Agree

We're going to work with an idea known as Agree.

- Assume first that finite verbs like missed bear a tense feature which determines their form (we'll talk about V and ignore *v* temporarily until we get a bit further along).
- Assume further that this feature must agree, i.e. match with the tense feature on T.
- This will ensure that the verb must have the form appropriate for the tense of the sentence determined by T.

This is a bit like the Checking requirement we used for selection:

- A selecting head has a dependent category feature, which checks off when matched with the independent version of the same feature on its sister.
- Here we also need matching, and we can imagine for starters that while the tense feature on T is independent, the one on the verbal complex is not.

Here's a first stab at defining it formally:

(3) Agree
A dependent feature F on a syntactic object Y is checked when another syntactic object Z bears a matching feature F.

☞ We're going to have to add something to this characterizing the structural relationship between Y and Z.

☞ But that will have to wait a few minutes until we've got a better idea of the empirical situation.

Now, since we're saying that the tense features on the verbal complex are dependent, this will ensure that they always match with those on T. Consider:

(4) T[past] . . . V [~~upast~~]

(5) T[past] . . . V [upres]

☞ The [upres] feature on V in 5 doesn't match the [past] feature on T, so Agree is not possible.

☞ This leaves [upres] unchecked, which will lead to a crash.

But how do we make sure that things actually come out right? We can imagine two scenarios:

1. The grammar combines V and T more or less randomly with all different possible feature values. The combinations that match, survive. Those that don't, crash.
2. The grammar is set up so that matching is actually derived by the derivation, not just checked for. Non-matching combinations simply never arise.

We tacitly assumed the first scenario in 4 and 5 above, but it is not entirely satisfying:

- ☞ It implies that we start up a whole bunch of derivations only to find out that we don’t have the right pieces.
- ☞ We end up throwing away potentially dozens of crashed derivations for every one that converges.

That sounds like a waste of computational resources.

So let's consider an alternative.

- Let's assume that V starts out with a tense feature that is not dependent, but unvalued.
- Agree then doesn't just check whether a dependent and an independent feature match, checking the dependent one.
- Instead, it checks whether the attributes of an unvalued and a valued feature match, and then values the unvalued one to be the same as the one it's agreeing with.

So we have:

(6) T[tense:past]. . . V[tense:] $\rightarrow$
T[tense:past]. . . V[tense:past]

A few things to note:

- We're using a special notation which distinguishes a feature's attribute (here tense) before the colon (:) from its value (here past) after the colon.
- Agree must now depend not on two features matching in their entirety, but on their attributes matching.
- Agree (involving unvalued features) is now quite distinct from Checking (involving dependent features) – perhaps a good thing since Agree doesn't seem to depend on sisterhood like Checking does.

Here's the new definition of Agree:

(7) Agree

In a configuration

X[F:val]. . . Y[F:]

The value from [F:val] is copied to [F:], resulting in:

X[F:val]. . . Y[F:val]

➪ Now situations like where V has the wrong tense will simply never arise.

C-command

Now that we have the basic idea of Agree, we need to come back to an important detail we skipped the first time around:

- ☞ As we've defined Agree, it should apply to any two syntactic objects bearing features with matching attributes.
- ☞ However, it's easy to show that this is not the case:

(8) * Tyrion thinks I lives in Leipzig.

(9) * I thinks Tyrion lives in Leipzig.

(10) * Tyrion was saying Danaerys living in the towers.

➪ So we're going to need a way to figure out what should Agree with what.

- ☞ This will mean putting conditions on the structural relationship between the two Agreeing objects.
- ☞ The condition of sisterhood used for Checking is too strict for Agree, so we'll need something new.
- ☞ The relation we'll adopt may seem a bit odd and abstract at first, but we'll see that it actually makes a lot of sense, and that it can do a lot of work for us.

(11) Node A c-commands node B iff A's sister either:

a. is B, or

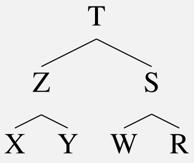
b. dominates B.

(12) Node C dominates node D iff

a. C is D's mother, or

b. C is the mother of a node E which dominates D

(13)



T c-commands nothing

Z c-commands S, W, R

X c-commands Y

Y c-commands X

S c-commands Z, X, Y

W c-commands R

R c-commands W

Here’s why c-command isn’t such a weird idea:

- ☞ An object will always c-command everything contained in the constituent it Merges with, and nothing else.
- ☞ Saying that syntactic operations like Agree are sensitive to c-command is essentially just saying that each object has one chance to make things happen.
- ☞ When it Merges with another object, it’s allowed to look inside it for instances of relevant features, but after that, it’s done.

So here's a new definition for Agree, with the addition in boldface:

(14) Agree
In a configuration
X[F:val]. . . Y[F:]

where . . . represents a c-command relation, the value from [F:val] is copied to [F:], resulting in:
X[F:val]. . . Y[F:val]

Reflexive binding

Now let's look at an empirical pattern that supports the idea that syntax cares about c-command. Consider the basics of the distribution of reflexive forms with -self:

(15) * Herself arrived.

(16) She kicked herself.

(17) * I kicked herself.

(18) * He kicked herself.

We can propose the following as a first shot at accounting for these facts:

(19) The Reflexive Generalization (version 1)
A reflexive pronoun must be coreferential with another expression in the sentence.

(20) The Coreferentiality Hypothesis
For two expressions to be coreferential, they must bear the same ϕ-features.

- We introduce the term ϕ-features to refer collectively to person, number and gender features.
- We need something like this to handle 18, where herself doesn't seem to be happy to go with he.

Now here's a contrast that 19 can't handle:

(21) I hit myself.

(22) * Myself hit me.

➪ It looks like the reflexive has to come after its antecedent.

So let's try this:

(23) The Reflexive Generalization (version 2)
A reflexive must be coreferential with a preceding expression in the same sentence.

But here are some more data that don’t fit with version 2 either:

(24) The girl I saw left.

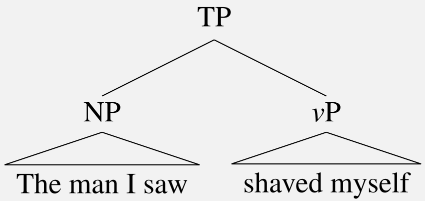
(25) * The girl I saw hit myself.

Consider some constituency facts:

(26) [She] left.

(27) It was [the girl I saw] who left.

So this should be the structure:

(28)


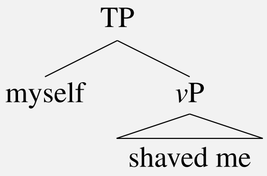
☞ Note that whatever the internal structure of that NP should be, I does not c-command myself.

So let's try this:

(29) The Reflexive Generalization (version 3)
A reflexive must be coreferential with a c-commanding expression in the same sentence.

This also covers the fact we started with:

(30)



And explains why possessors generally can't bind:

(31) * [My mother] hated myself.

We're still a long way from getting reflexive binding right – for one thing the sentence is not the right-sized domain for binding.

☞ But whatever the precise characterization of binding domains and requirements turns out to be, it is clear that c-command will play a crucial rule.

So reflexives provide some empirical evidence that c-command is relevant for grammar, thus it's not a bad idea to include it in our definition of Agree.

- ? But so now how can we try to make our new reflexive generalization fit in with the rest of the theory we're building?
- ☞ As it stands right now, the reflexive generalization is a completely independent principle, not integrated with anything else.
- ☞ It has c-command in common with Agree, but this is just a stipulation, not derived in any principled way.

Now, note interestingly enough that the phenomenon of reflexives fits into the broader category we mentioned of action at a distance:

- ☞ The form and reference of the reflexive element depends on the NP that binds it.
- ➪ So clearly our strategy should be to try to reformulate the reflexive generalization in terms of Agree, our new operation for dealing with such phenomena.

There are a number of Agree-based approaches that have been proposed for binding in recent years.

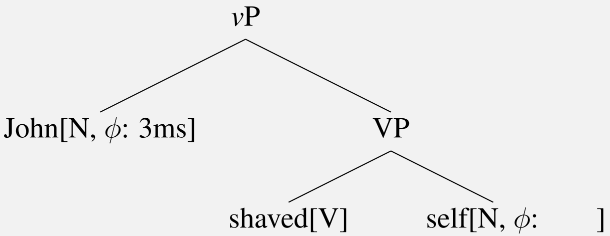
- We’re going to adopt one here which is quite popular, even though we’re pretty sure it will turn out to be wrong in some important details.
- Still, it has the advantages of getting the basics right, and of being relatively straightforward – not requiring us to add too many exotic features and things to our theory.

The basic idea is that binding is actually Agree for ϕ-features.

- ϕ-features are the features normally involved in agreement (which we'll discuss later on).
- And again, they're precisely the features that must match between a reflexive and its binder, so it is natural to think they are somehow involved in an Agree relation.
- All that has to be said in addition then is that the defining property of reflexives is that they come into the derivation with unvalued ϕ-features.
- The reflexive generalization then just falls out of the fact that these features will have to get valued, via Agree with a c-commanding NP with valued ϕ-features.

Here's how it would look:

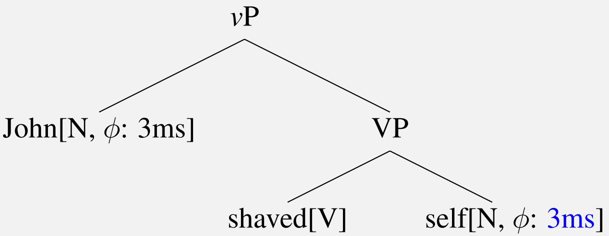
(32)



(Actually, there is one open issue related to the direction of the c-command relationship which we'll set aside for now...)

Here's how it would look:

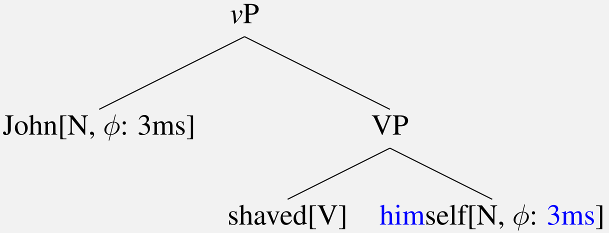
(32)



(Actually, there is one open issue related to the direction of the c-command relationship which we'll set aside for now. . .)

Here's how it would look:

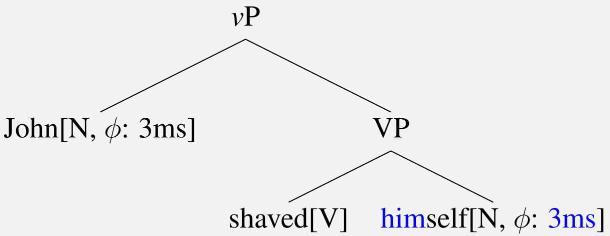
(32)



(Actually, there is one open issue related to the direction of the c-command relationship which we'll set aside for now…)

Here's how it would look:

(32)



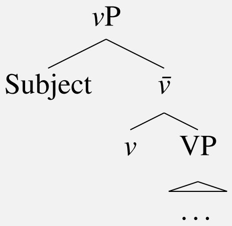
(Actually, there is one open issue related to the direction of the c-command relationship which we'll set aside for now…)

Introducing Move

Now we turn to a different kind of action at a distance, where an entire syntactic object shows up where we don't expect it. We start with a word-order issue mentioned last time:

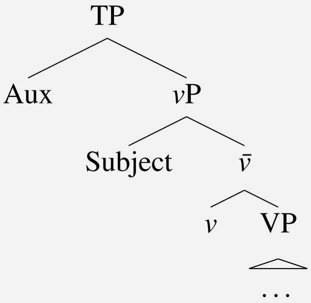
- Recall that, for thematic reasons, we think that subjects are merged in Spec-*v*P:

(33)



- And we’ve argued that finite auxiliaries surface in T, above *v*P:

(34)



This predicts that subjects will come after the finite auxiliary, but of couse they come before it, at least in declaratives:

(35) * Will Cassandra foretell disaster again.

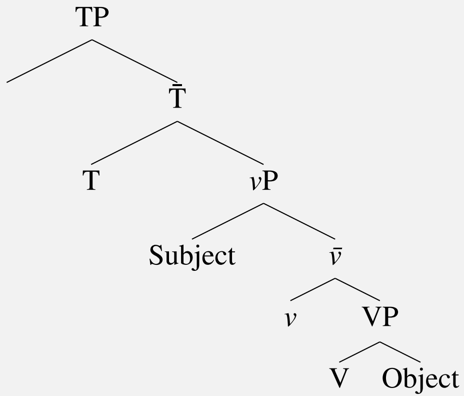
(36) Cassandra will foretell disaster again.

We are forced to find a new way to deal with this.

- What we're going to say is that the subject does initially Merge in Spec-*v*P, as expected on the basis of θ-roles.
- Then we'll say that it Moves to Spec-TP.

Here's how it will look:

(37)



☞ We indicate the place where an element has moved from with a copy of the element formatted like <this>.

Here's how it will look:

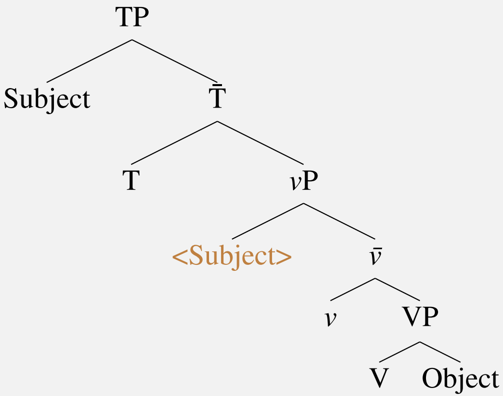
(37)

$$\begin{tikzcd}[column sep=small]
 & \text{TP} \arrow[dl, dash] \arrow[dr, dash] & & & & \\
\text{Subject} & & \bar{\text{T}} \arrow[dl, dash] \arrow[dr, dash] & & & \\
 & \text{T} & & v\text{P} \arrow[dl, dash] \arrow[dr, dash] & & \\
 & & \text{<Subject>} & & \bar{v} \arrow[dl, dash] \arrow[dr, dash] & \\
 & & & v & & \text{VP} \arrow[dl, dash] \arrow[dr, dash] \\
 & & & & \text{V} & & \text{Object}
\end{tikzcd}$$

☞ We indicate the place where an element has moved from with a copy of the element formatted like <this>.

Here's how it will look:

(37)



☞ We indicate the place where an element has moved from with a copy of the element formatted like <this>.

Again, this new operation Move represents a serious addition to the complexity of our theory, so we're going to need to think very carefully about it and make sure it's justified.

- ? Is there any additional evidence for the subject movement?
- ? Are there any other examples of similar movements?
- ? Can we explain why this movement should happen?
- ? Can we integrate Move somehow with the rest of our theory in a way that keeps overall complexity to a minimum?

Evidence for subject movement

Quantifier float

Consider sentences with quantified subjects:

(38) All the dragons had escaped.
(39) Both the twins might have been at the party.

These also have paraphrases where the quantifier ‘floats’ away from the subject:

(40) The dragons had all escaped.
(41) The twins might both have been at the party.

How can we explain this behavior? How about this:

☞ The quantifiers start out with the NPs, but the NPs move to the left, and the quantifiers don't always come along.

Specifically:

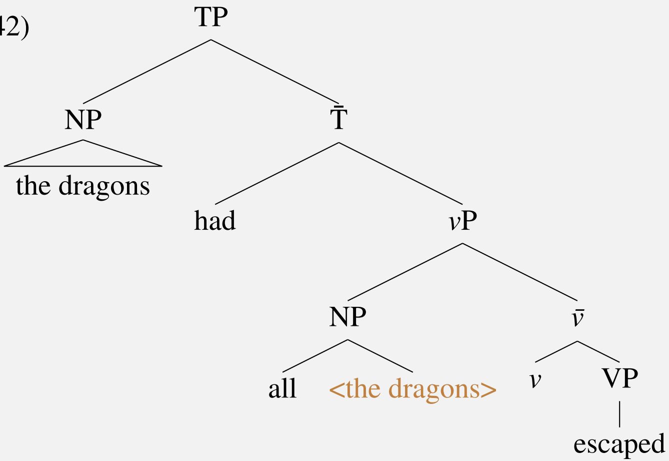
- The Q+NP combo starts out in Spec-*v*P, and the NP subsequently moves to TP.
- Sometimes the Q comes along for the ride, and sometimes it doesn't.

This has some promise:

- ☞ It captures the fact that in both orders, the quantifiers semantically seem to relate to the NPs.
- ☞ And the movement we have to assume is independently supported, given the order of subjects and auxiliaries.

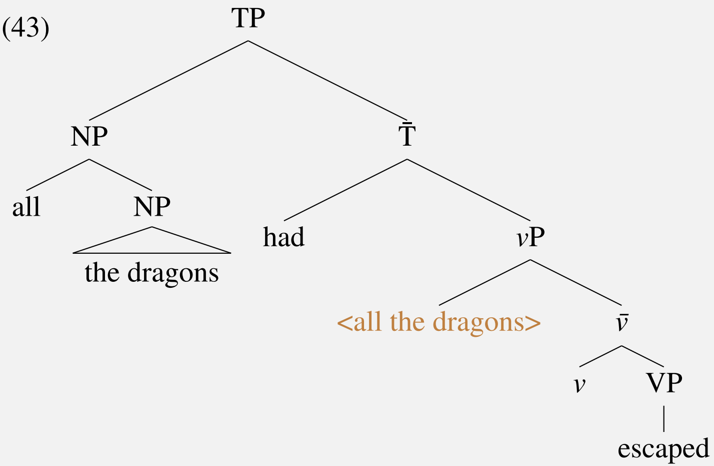
So here's how it would look if just the NP moves:

(42)



And here’s the version where the quantifier goes along:

(43)



Expletives

There's a special use of the word there, which isn't about location, and in fact seems to have no real meaning:

(44) There are many fish in the sea.

(45) There were people playing on the beach.

- there is called an expletive, which essentially means placeholder.
- In these sentences, there is holding the place of the subject.

So note that there are similar sentences where the normal subject occupies the position that was held by there:

(46) Many fish are in the sea.
(47) People were playing on the beach.

And note that there behaves like the subject for purposes of tag-question formation:

(48) People are playing on the beach, aren't they?
(49) There are people playing on the beach, aren't there?

This furnishes another argument for subject movement:

☞ Expletive sentences have the following form:

(50) there T … subject *v*P

☞ Parallel sentences without expletives appear to have this form:

(51) subject T … *v*P

☞ The surface position of the subject with expletives makes a lot more sense if what underlies 51 is really this:

(52) subject T … <subject> *v*P

- ➪ So we have good evidence that subjects start out in a lower position, which we call Spec-*v*P and move to a higher position, which we call Spec-TP.
- ☞ Now we need to motivate this movement within our theory, and ideally explain why it happens.

We'll work on that, along with other aspects of movement, in the next handout.

Introduction

Our job now is to explore this new operation Move.

- Can we further justify the addition of this operation to our theory with new data?
- Why does language make use of Move, and why do particular items Move while others don't?
- How can we integrate Move into our existing theory so as to keep things as simple as possible?

Move as Merge

Taking the last question first, we start with the observation that Move has a lot in common with our other operation Merge.

- At a very simple level, both affect the positioning of elements in the hierarchical structure.
- In fact, if we think of Move as taking something in the structure and combining it with a different part of the structure, both operations involve combining things.

Of course, there is a clear asymmetry here:

☞ Merge seems to be simpler than Move, since it's really just the combining step.

Merge is also indispensable in a way that Move is not:

- The essence of a syntax is the combination of simpler pieces to make more complex structures, so something like Merge must be part of the system.
- This is not just true of human language syntax but of the syntax of any symbolic system. E.g. formal logic, programming languages and recipes all involve a counterpart of Merge.
- It is not clear however that something like Move should be needed. We can perfectly well imagine a symbolic system that does not have anything that looks like it.
- So e.g. there is no obvious analog to Move in the non-language systems just mentioned.

All of this leads to the conclusion that we should try to explicitly relate Move to Merge within our theory.

➪ To the extent possible, we should in fact redefine Move in terms of Merge.

Move is built on Merge

Let's start with the idea mentioned above, that Move involves taking something within the existing structure, and combining it with a different part of the structure.

- The obvious thing to do now is to make that second combining step actually be Merge.

(53) Move (first version): Take a syntactic object that is a constituent of an existing structure and Merge it elsewhere.

There are two things we need to do then:

1. Make sure that the second part of Move really can conform to our existing definition of Merge.
2. Make more precise the first step of grabbing a syntactic object to Merge elsewhere.

Recall our definition of Merge:

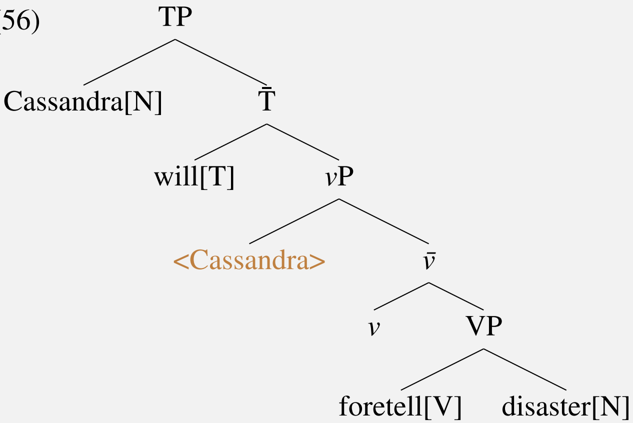
(54) Merge: Take two syntactic objects, and join them together at their roots to form a new syntactic object.

So the only thing we really have to worry about here is that when something Moves, it should Move to the root of the existing structure, since we want to continue to respect the Extension Condition.

(55) Move (second version): Take a syntactic object that is a constituent of an existing structure and Merge it with the entire structure.

For the moment, this doesn't present any problems, since the only instance of Move we've posited so far obeys this:

(56)



- When the subject Cassandra Moves, it Merges with the node labeled $\bar{\text{T}}$, which is the root node at that point in the derivation.
- In more advanced versions of syntax, you will see that things aren't always so simple, but this works for now.