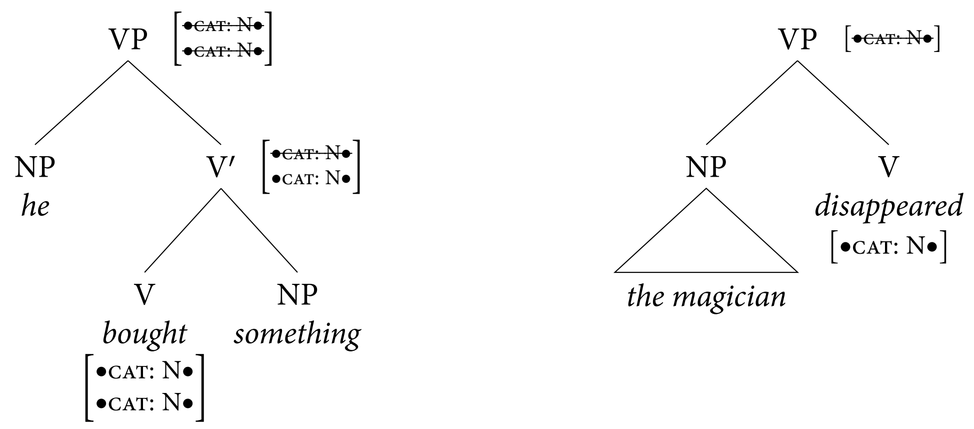
Lecture notes: Week 4
Ditransitives and adjuncts

1 Ditransitive verbs

- So far, we have talked about how we can build structures with Merge.
- This means that all structure building involves combinations of two things. The only structures we can build with Merge are **binary branching** (where every non-terminal node has exactly two daughters).
- Recall that we had identified three different types of rules we need for VP, each corresponding to a different verb type:

(1) a. VP → Vi (intransitive VP rule)
b. VP → Vt NP (transitive VP rule)
c. VP → Vd NP NP (ditransitive VP rule)

- Last time, we saw how the new Merge-based approach can derive transitive verbs and intransitive verbs (albeit with a somewhat unresolved issue about word order, which will come back to).

(2)


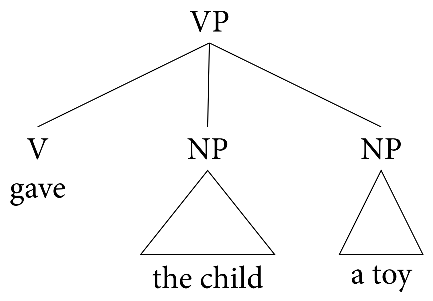
- Now, we will turn our attention to ditransitive verbs.
- Let's take the example in (3).

(3) I gave the child a toy

- It will be useful to have some terminology to distinguish the two objects of a ditransitive verb:
- The first argument *the child* is the **indirect object** of the verb. In terms of its meaning, it is the receiver of the giving action, but not the thing being given.
- The **direct object** of the verb is the thing that is given, in this case this is *a toy*.

- The rule in (1) gives us a Structure A like (4), but we have seen that it is not possible to build such a tree with our definition of Merge.

(4) *Structure A*

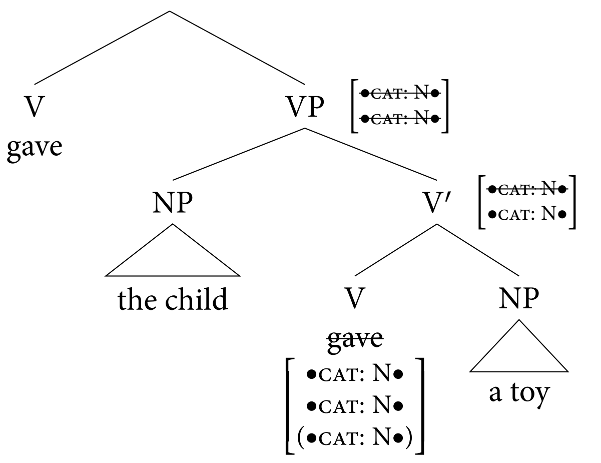


- But then again, our theory of Merge could be wrong, so let's keep this structure in my mind for now.
- Since a verb like *gave* combines with three NPs (a subject and two objects), it needs to have three selectional features [•CAT: N•].

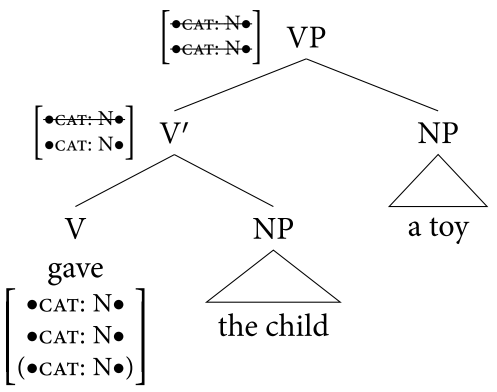
(5) V
gave
$\begin{bmatrix} \bullet\textsc{cat}\text{: N}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet \\ (\bullet\textsc{cat}\text{: N}\bullet) \end{bmatrix}$

- For now, we will ignore the third selectional feature for the subject and just focus on the order in which the objects are merged.
- There are two possibilities for this: either the direct object *a toy* is merged first as the complement of *give*, as in Structure B below, or the the indirect object *the child* is merged first, as in Structure C.

(6) *Structure B*



(7) *Structure C*



- Neither of the structures reflect the word order we want straightforwardly, however.

- In (7), we have to assume some kind of displacement of the verb in order to get the right word order.
- In (6), we do not need displacement of the verb if we assume that a specifier can be pronounced to the right of its sister constituent.
- Since both of these structures are plausible representations of the word order of the English ditransitive, how do we know which one is the right one?
- In order to do this, we need to appeal to a syntactic diagnostic that we introduced last time: **c-command**.

C-command

X c-commands Y if…

a. Y is the sister of X

or

b. Y is dominated by X

- Structures B and C have different relations between the two objects of the verb.
- In Structure B (6), the indirect object (*the child*) c-commands the direct object (*a toy*).
- In Structure C (7), the direct object c-commands the indirect object.
- In order to distinguish between these approaches, we need to have tests that show us which c-command relation is the right one.
- Recall from the previous notes, that pronouns containing *-self* have a special requirement which we can summarize in the *self*-rule.

***self*-rule**

A pronoun containing *self* must refer to the closest c-commanding matching referent.

- This accounts for the data in (8).

(8) a. Steve made Jake embarrass himself at the party.
 b. Steve made himself embarrass Jake at the party.
 c. $\left\{\begin{array}{l}\text{A friend of Steve}\\\text{A friend of Steve}\end{array}\right\}$ embarrassed himself at the party.

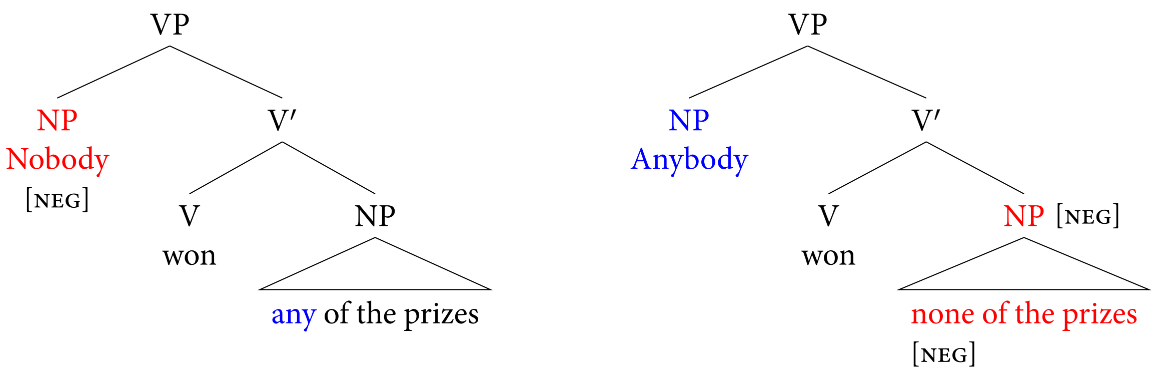
- Let us now consider another restriction similar to *self*-pronouns.
- This will come from the word *any*. This word shows a particular dependence on negation.[1]

[1]This is a significant simplification, but it will work for present purposes. *Any* and some words with similar properties are called *negative polarity items*.

- As the following sentences illustrate, *any(body)* is ungrammatical if it is not preceded by a negative expression such as *nobody*.

(9) a. Nobody won all of the prizes
b. Nobody won any of the prizes
c. I won any of the prizes
d. *Anybody won all of the prizes
e. *Anybody won none of the prizes

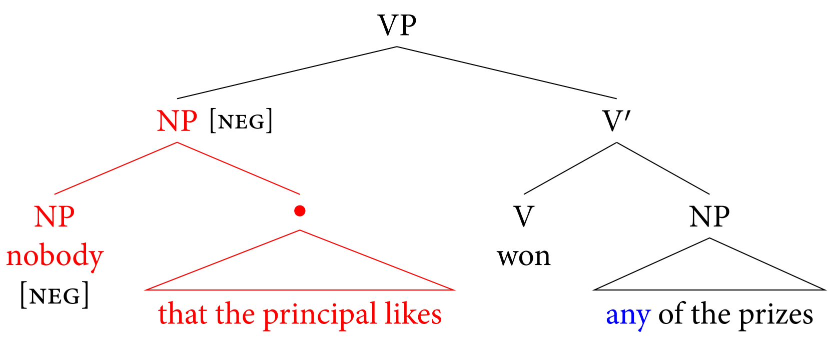
- While we might be tempted to appeal to a linear restriction here, the following sentences show that it is actually about c-command (similar to the argument we saw with *self*-pronouns), as shown below.

(10)


- This view if further supported by the following sentences:

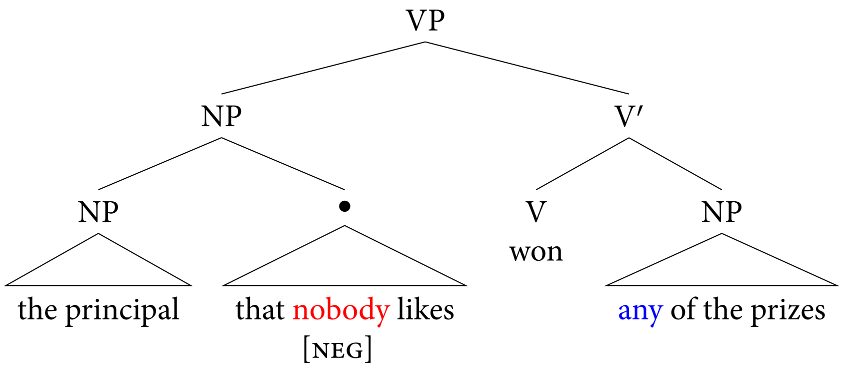
(11) a. Nobody that the principal likes won all of the prizes
b. Nobody that the principal likes won any of the prizes
c. The principal that nobody likes won all of the prizes
d. *The principal that nobody likes won any of the prizes

- When the subject is the negative phrase *nobody that the principle likes*, then it c-commands any in the object *any*.
- (I am assuming that *no(body)* bears a feature [NEG] that is projected to the full NP.)

(12)


- This is not the case when the negative expression is embedded in the subject (there is good reason to assume that the [NEG] does not project out of the • constituent here).

(13)



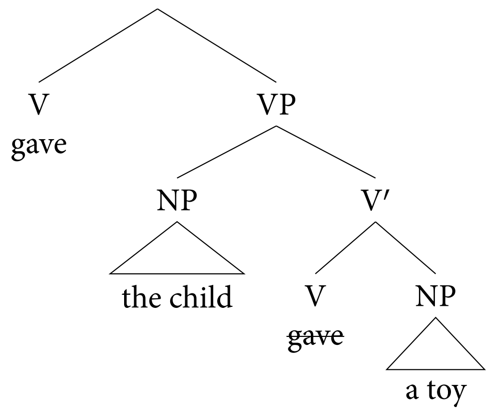
- We can therefore posit the following rule for *any*:

***any*-rule**

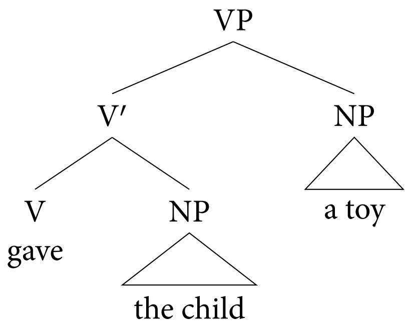
A noun phrase containing *any* must be c-commanded by a negative phrase.

- Now returning to ditransitives, recall the two structures we are trying to distinguish.

(14) *Structure B*



(15) *Structure C*

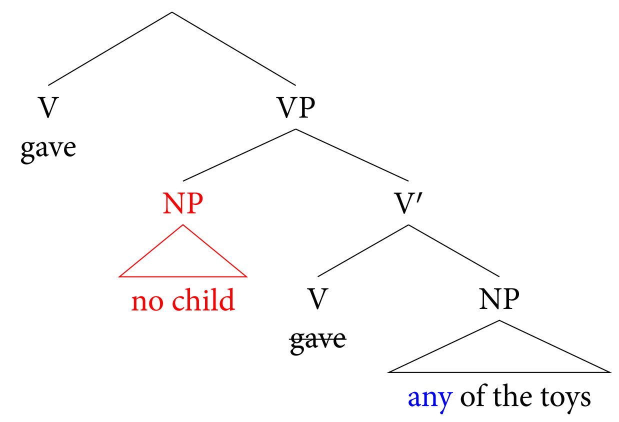


- First, let's try making the indirect object negative and putting *any* in the direct object:

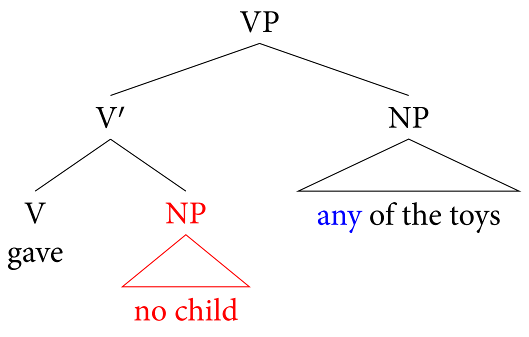
(16) I gave no child any of the toys

- This sentence is acceptable, so this must mean that that *any* is c-commanded by a negative phrase.
- This is true in Structure B, but not in Structure C.

(17) *Structure B*



(18) *Structure C*

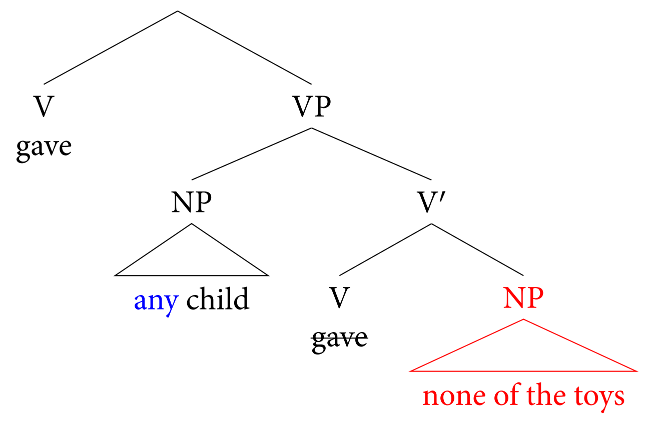


- This an argument in favour of Structure B.
- Now let's reverse the positions and put *any* in the indirect object and make the direct object a negative phrase. This sentence is ungrammatical:

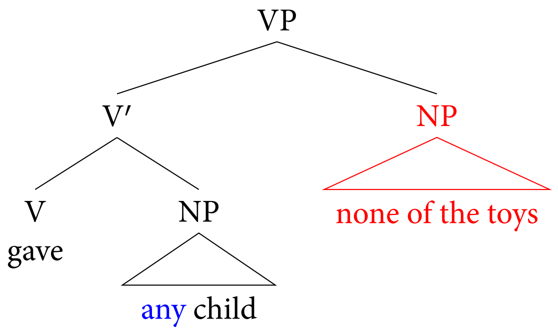
(19) *I gave any child none of the toys

- This means that the correct structure should be one in which *any* is not c-commanded by a negative NP.

(20) *Structure B*



(21) *Structure C*



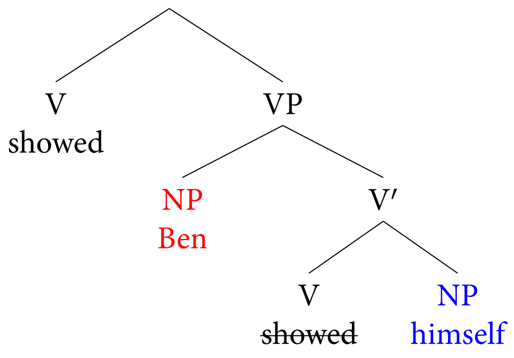
- In Structure C, *any* is c-commanded by a negative phrase, so this makes the wrong prediction here. The lack of c-command in Structure B means that this is another argument in favour of this structure.
- So, the *any*-rule gives us two arguments for Structure B (and against Structure C).
- Now, let's return to pronouns with *self*. We can run the same tests here. Since both the indirect and direct object have to be animate/human, it is a little strange to use the verb *give* (we don't normally think about people being the object of giving).

- Instead, let's use the verb *show* and imagine a situation where someone is pointing out the reflection of someone in the mirror (you might think of being at the hairdresser as a plausible scenario for this).

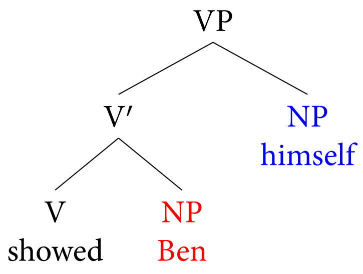
(22) She showed Ben himself (in the mirror).

- Since this sentence is grammatical, the correct sturcture will be one in which *himself* is c-commanded by the closest matching referent.

(23) *Structure B*



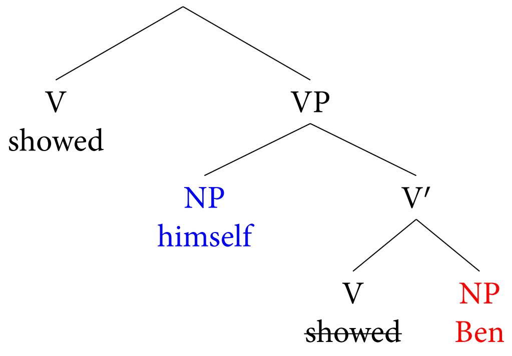
(24) *Structure C*



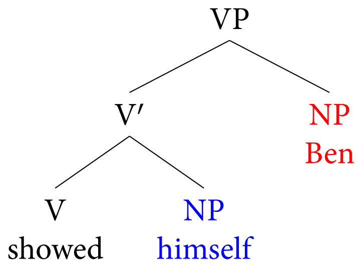
- Again, this is a correct prediction for Structure B, but not for Structure C.
- Let's switch the positions to make the *self*-pronoun the indirect object:

(25) *She showed himself Ben (in the mirror).

(26) *Structure B*

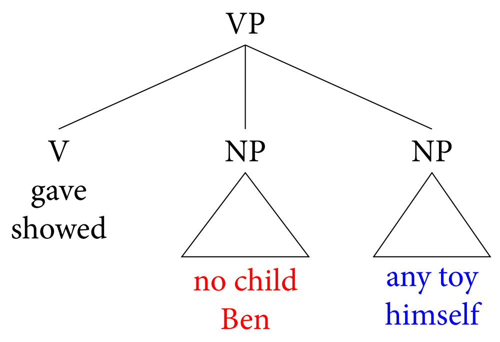


(27) *Structure C*



- Structure C predicts this sentence to be grammatical, as *himself* is c-commanded by the matching antecedent *Ben*. This an incorrect prediction, of course. The lack of c-command of *himself* by *Ben* in Structure B is compatible with the ungrammaticality of this example.
- So, to take stock: We have looked at four sentences to test the c-command relations in a ditransitive VP and all of them point in favour of Structure B. For this reason, let's disregard Structure C.
- What about Structure A?

(28) *Structure A*



- As we see above, Structure A predicts no asymmetry between the indirect and direct object, as they c-command each other. It correctly predicts the grammatical sentences (e.g. *She showed Ben himself*), but it equally predicts the ungrammatical sentences to be possible (e.g. **She showed himself Ben*). For this reason, we can also disregard Structure A.

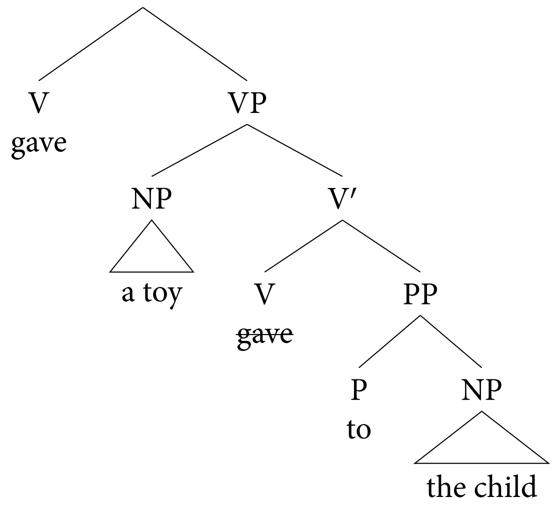
- Assuming Structure B is correct, let's now consider another type of ditransitive. In addition to sentences like (29a), we can also say (29b).

(29) a. I gave the child a toy
 b. I gave a toy to the child

- This is different as the indirect object in (29a) is actually realized inside a prepositional phrase (PP) as the sister of *to*.

- Our c-command tests above will also converge on this as the right structure for such sentences (feel free to check for yourself!):

(30)



- This variant is called the *prepositional dative construction*, whereas (29a) is called the *double object construction*.

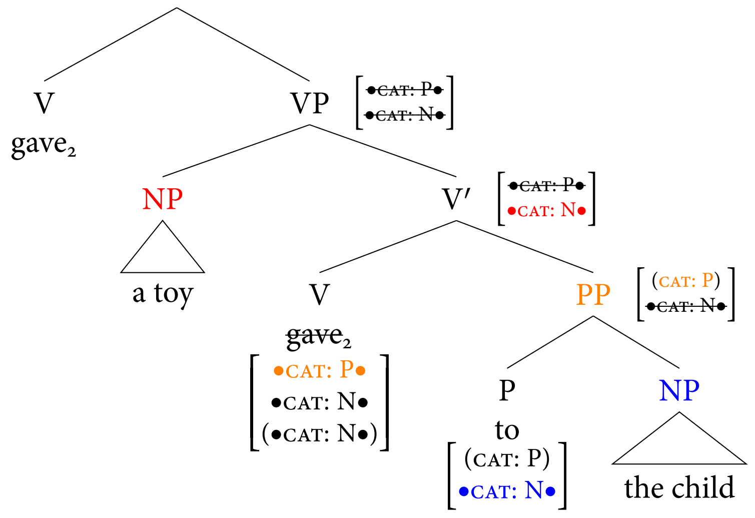
- In terms of features, it is clear that we will need to have a different selection requirements. The lexical item *gave* will therefore have to be able to bear features selecting a phrase of category P and also two phrases of category N.

- We can distinguish this by assuming that *gave*$_1$ and *gave*$_2$ are separate entries in the lexicon:

(31)
$$\underset{\begin{bmatrix}\bullet\textsc{cat: N}\bullet\\ \bullet\textsc{cat: N}\bullet\\ (\bullet\textsc{cat: N}\bullet)\end{bmatrix}}{\overset{\text{V}}{\textit{gave}_1}} \qquad \underset{\begin{bmatrix}\bullet\textsc{cat: P}\bullet\\ \bullet\textsc{cat: N}\bullet\\ (\bullet\textsc{cat: N}\bullet)\end{bmatrix}}{\overset{\text{V}}{\textit{gave}_2}}$$

- If we choose to the build a sentence as a projection of *gave*$_2$, then we get the following structure:

(32)

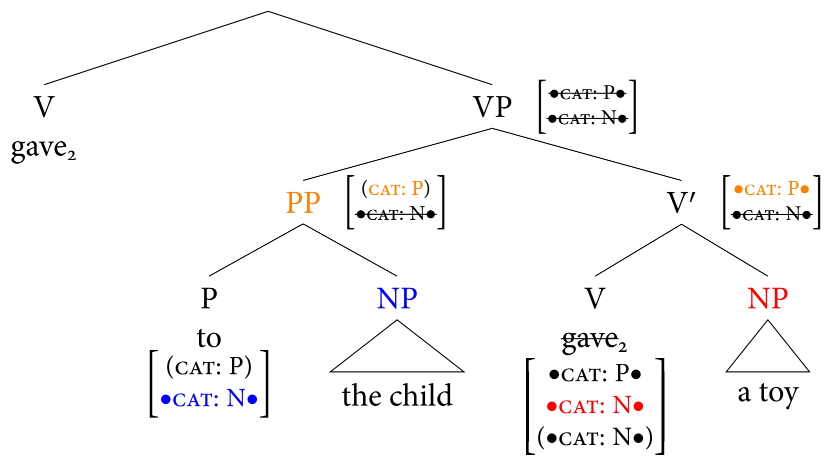


- An important question that comes up here, however, is why we can't merge the arguments of *gave* in the opposite order as in (33). This would give us **I gave to the child a toy*. This is not something we want.[2]

[2]You might think this sounds OK in some contexts (e.g. if *a toy* is strongly emphasized), but this almost certainly due to a different process that we will come to soon (rightward displacement).
The fact that we don't generally have the kind of optionality we would get from allowing (33) is shown by the contrast below:

(i) a. I introduced Jane to her new boss
b. *I introduced to her new boss Jane

(33)

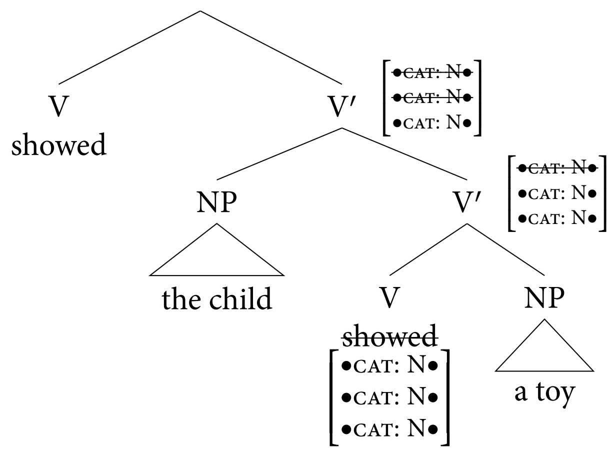


- What is the solution here? Let's assume that the set of features is a **stack**, such that only the top-most (unchecked) feature is available to be checked on that stack. In the arrangement of the stack in (33), we actually cannot merge an NP before a PP because [•P•] is on top of the stack.
- (Recall that these features are actually deleted, so they aren't on top of the stack anymore after that.)
- This assumption means that the order of the merge operations is actually fixed by the way we list the features.

2 The position of the verb and the subject in VP

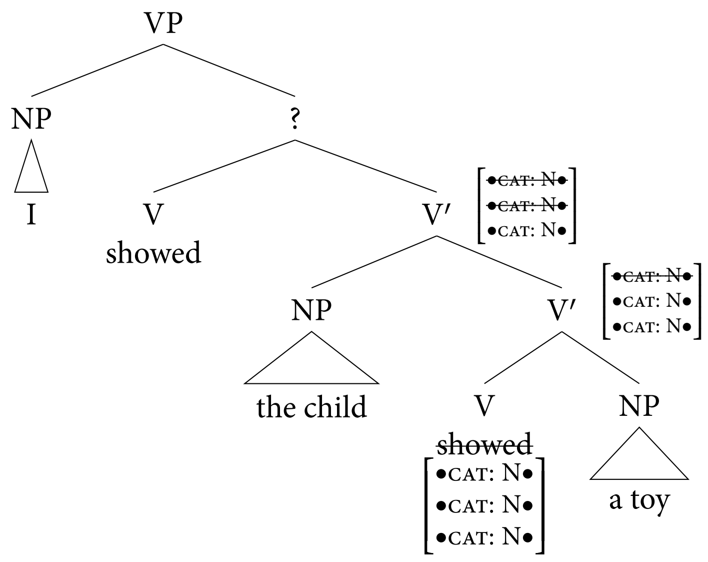
- Now that we have arrived at the correct c-command relations between the indirect and direct object, let's return to the question of where the subject comes in.
- Given the structure in (34), we still have to find a way to check the third remaining selectional feature of *showed*:

(34)



- If we merge the subject with the constituent created after displacement of *showed*, then the local sisterhood relation between V′ and the NP *I* is not given.

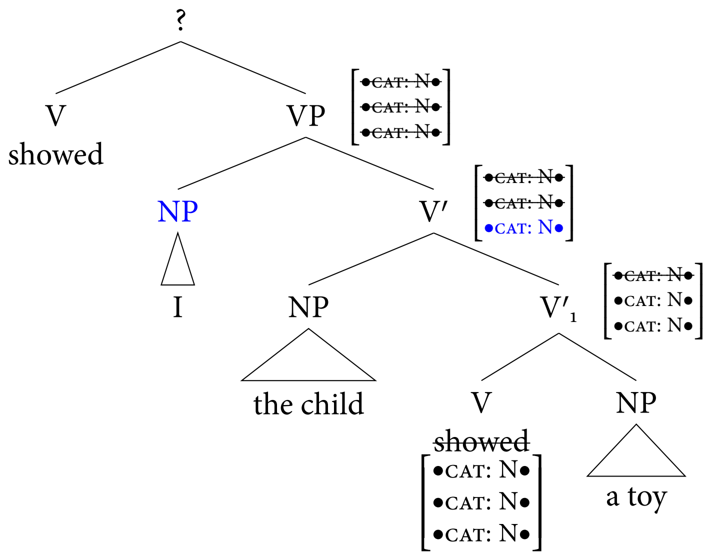
(35)



- (An option that we won't follow here is that the verb contains a feature like [•CAT:V•] that would trigger displacement of the verb itself to V′. The features we have mentioned here trigger merge of phrases, not heads. We will see that we need a different kind of operation to move heads and that this displacement has different properties.)

- The other option is to make the subject the sister to V′. This solves our feature checking problem, but now the word order issue is back.

(36)

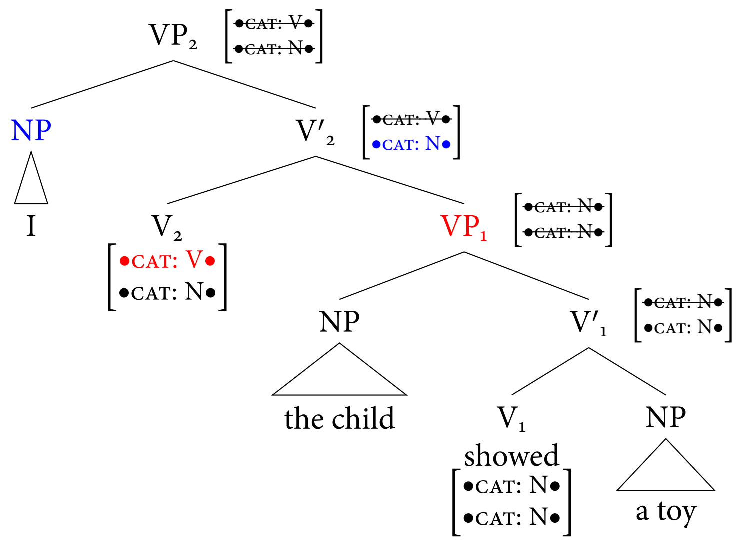


- The solution I would like to propose is that the subject is not selected by the verb *gave* at all.

- If we think about the types of verbs we have (intransitive, transitive and ditransitive), they all have in common that they have a subject. In fact this is why *showed* (in the sentences above) is a 'ditransitive' and not a 'tritransitive' verb, since the name refers to the number of objects (where 'transitive' means something like 'has an object').

- Let's then assume that a verb only has selectional features for its objects and 'outsource' the job of the selecting the subject to another head. I will call this head V_2 for now, as in the tree below.

(37)



VP₂ [~~•CAT: V•~~ ~~•CAT: N•~~]
NP
I
V′₂ [~~•CAT: V•~~ •CAT: N•]
V₂ [•CAT: V• •CAT: N•]
VP₁ [~~•CAT: N•~~ ~~•CAT: N•~~]
NP
the child
V′₁ [~~•CAT: N•~~ •CAT: N•]
V₁
showed [•CAT: N• •CAT: N•]
NP
a toy

- You might be wondering how this fixes our word order problem. Well, it doesn't unless we displace the verb as we did before.

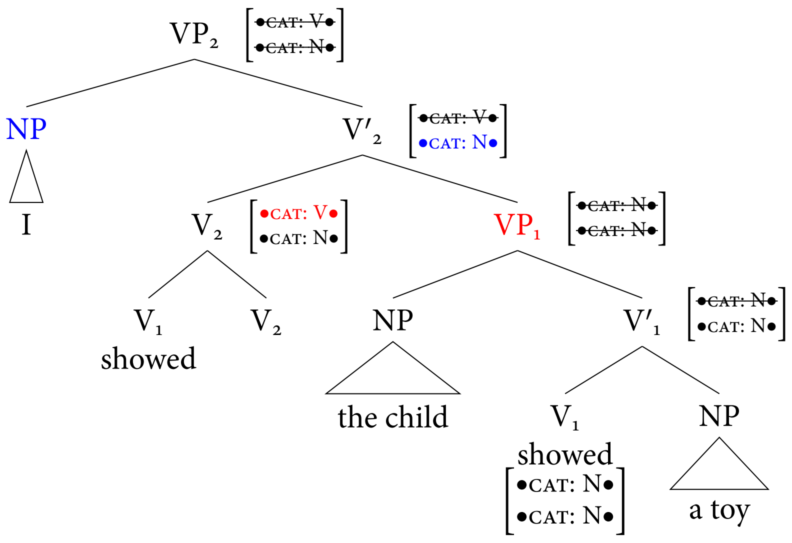
- It seems that verb has to move the position that V_2 is in. In order to do this, we are going to need a different kind of displacement operation. Since Merge only allows us to target the root node of the tree, what we want is a displacement operation that allows *showed* to occupy the position that V_2 does.

- We will call this operation **head movement** (movement of a head to another head position). Let's define it as follows for the time being:

Head movement

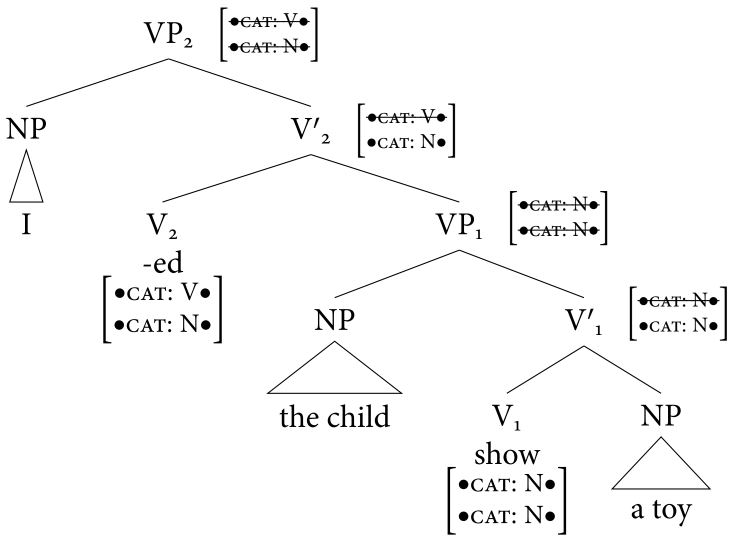
Create a new node Z with daughters X and Y, where X is a minimal projection (≈head) and Y is the closest c-commanding head to X. Do not pronounce the lower occurrence of X. Also, Z = Y.

(38)



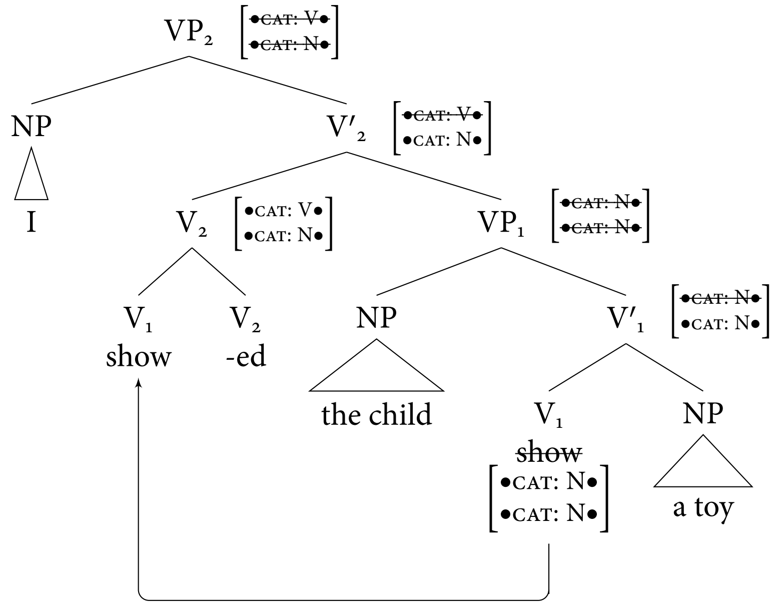
- At this point, you might be wondering what V_2 is and why we need it (other than to create a place for the verb to move to).
- Well, it turns out that having this additional head V_2 will be useful for a number of reasons.
- The first one is that we can assume that this is place where the regular morphological endings (e.g. *-ed*, *-s*) on the verb are hosted in the tree.
- On this revised view, the structure actually looks like this:

(39)



- Now, the verb that selects the arguments is just *show*, not *showed*. In moving to V_2, it combines with its *-ed* suffix as a result of head movement (while still getting its position in the sentence right relative to the objects):

(40)



VP₂ [~~•CAT: V•~~ ~~•CAT: N•~~]
NP
I
V′₂ [~~•CAT: V•~~ •CAT: N•]
V₂ [•CAT: V• •CAT: N•]
VP₁ [~~•CAT: N•~~ ~~•CAT: N•~~]
V₁ show
V₂ -ed
NP the child
V′₁ [~~•CAT: N•~~ •CAT: N•]
V₁ ~~show~~ [•CAT: N• •CAT: N•]
NP a toy

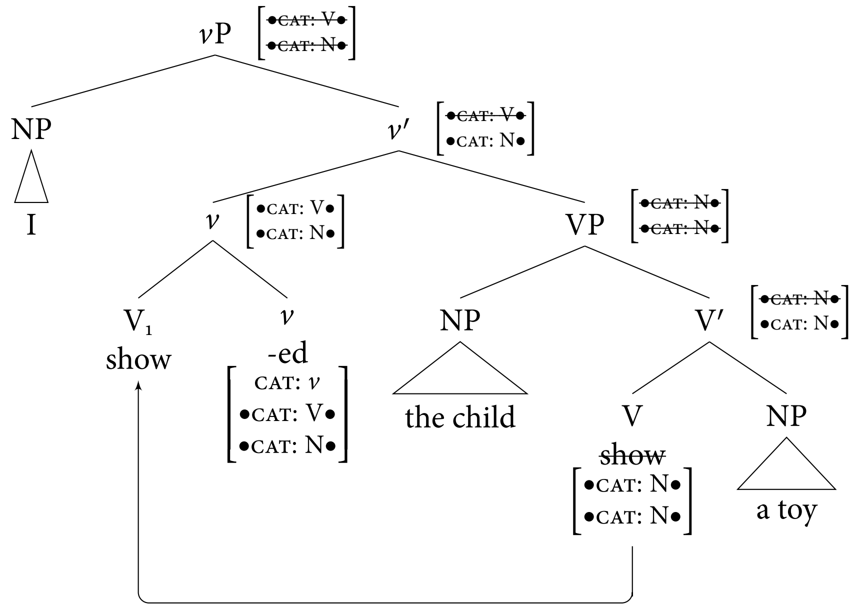
- While *show* is a **lexical item** and is associated with a word, we can call V_2 a **functional item** (or a **functional head**). It can be overtly realized (e.g. as *-ed*), but functional items need not have any realization. They instead fulfil a certain role or function in our theory. In this case, the function of V_2 is to host the regular morphology associated with the verb. (We will see more functions of V_2 in the coming weeks.)

- Since this is a different kind of head to the lexical verb, it seems wrong to call it V_2. This implies it is another instance of the same kind of thing as V_1, the actual verb.

- Instead, let's rename V_2 as *v*, pronounced 'little V'. (This was the name given to it by Noam Chomsky.)

- With this change in place, the tree in (40) becomes the one in (41).

(41)



vP [~~•CAT: V•~~ ~~•CAT: N•~~]
NP
I
v′ [~~•CAT: V•~~ •CAT: N•]
v [•CAT: V• •CAT: N•]
VP [~~•CAT: N•~~ ~~•CAT: N•~~]
V₁ show
v -ed [CAT: v •CAT: V• •CAT: N•]
NP the child
V′ [~~•CAT: N•~~ •CAT: N•]
V ~~show~~ [•CAT: N• •CAT: N•]
NP a toy

- (The only change is replacing V_2 with v. We no longer need to call the lexical projection of the verb V_1 to distinguish it. Also, I have added the category feature to v just to make clear that this is a different kind of category. It is a **functional category**.)
- Separating the morphology from the verb has some conceptual advantages, too.
- In a world where both *show* and *showed* are listed in the lexicon, then we unnecessarily duplicate certain information about these verbs. For example, imagine our lexicon only contains the simple present and past forms of *show* and *give*. Since each of these verbs can appear in the prepositional dative and the double object construction, we need to list two variants ($\textit{show}_1$ and $\textit{show}_2$) in order to encode this.
- If morphological variation is taken into account, then all forms of each of the two verbs must be listed in addition.
- (This is also true for the forms we aren't considering yet like *shown*, *showing*, etc. To distinguish *give* and *gives*, we need another feature that we haven't talked about it.)
- These leads to massive redundancy in the lexicon:

Redundant lexicon

$$
\begin{array}{cccccc}
\text{V} & \text{V} & \text{V} & \text{V} & \text{V} & \text{V} \\
\textit{show}_1 & \textit{shows}_1 & \textit{showed}_1 & \textit{show}_2 & \textit{shows}_2 & \textit{showed}_2 \\
\begin{bmatrix}\bullet\textsc{cat}\text{: N}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet \\ \textsc{tense}\text{: }\textsc{pres}\end{bmatrix} &
\begin{bmatrix}\bullet\textsc{cat}\text{: N}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet \\ \textsc{tense}\text{: }\textsc{pres}\end{bmatrix} &
\begin{bmatrix}\bullet\textsc{cat}\text{: N}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet \\ \textsc{tense}\text{: }\textsc{past}\end{bmatrix} &
\begin{bmatrix}\bullet\textsc{cat}\text{: P}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet \\ \textsc{tense}\text{: }\textsc{pres}\end{bmatrix} &
\begin{bmatrix}\bullet\textsc{cat}\text{: P}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet \\ \textsc{tense}\text{: }\textsc{pres}\end{bmatrix} &
\begin{bmatrix}\bullet\textsc{cat}\text{: P}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet \\ \textsc{tense}\text{: }\textsc{past}\end{bmatrix} \\
\text{V} & \text{V} & \text{V} & \text{V} & \text{V} & \text{V} \\
\textit{give}_1 & \textit{gives}_1 & \textit{gave}_1 & \textit{give}_2 & \textit{gives}_2 & \textit{gave}_2 \\
\begin{bmatrix}\bullet\textsc{cat}\text{: N}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet \\ \textsc{tense}\text{: }\textsc{pres}\end{bmatrix} &
\begin{bmatrix}\bullet\textsc{cat}\text{: N}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet \\ \textsc{tense}\text{: }\textsc{pres}\end{bmatrix} &
\begin{bmatrix}\bullet\textsc{cat}\text{: N}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet \\ \textsc{tense}\text{: }\textsc{past}\end{bmatrix} &
\begin{bmatrix}\bullet\textsc{cat}\text{: P}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet \\ \textsc{tense}\text{: }\textsc{pres}\end{bmatrix} &
\begin{bmatrix}\bullet\textsc{cat}\text{: P}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet \\ \textsc{tense}\text{: }\textsc{pres}\end{bmatrix} &
\begin{bmatrix}\bullet\textsc{cat}\text{: P}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet \\ \textsc{tense}\text{: }\textsc{past}\end{bmatrix}
\end{array}
$$

- What we are encoding here is that these verbs may appear in two different syntactic constructions ($\textit{show}_1$ vs. *show*2), but also that they have different forms depending on the tense specification of sentence (e.g. present tense vs. paste tense).
- With a functional element like v in our lexicon, we can place the feature responsible for the form of the verb (in this case, tense) on that head.
- This allows us to stick to two basic forms of each verb which simply encode their selectional properties. The head v carries the relevant information about tense, either the value PAST or PRES(ENT) (we will see how it gets this next week).

Non-redundant lexicon

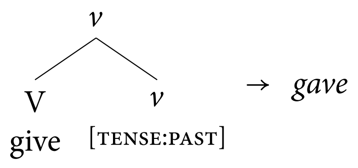
$$
\begin{array}{ccccc}
\text{V} & \text{V} & \text{V} & \text{V} & v \\
\textit{show}_1 & \textit{show}_2 & \textit{give}_1 & \textit{give}_2 & \\
\begin{bmatrix}\bullet\textsc{cat}\text{: N}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet\end{bmatrix} &
\begin{bmatrix}\bullet\textsc{cat}\text{: P}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet\end{bmatrix} &
\begin{bmatrix}\bullet\textsc{cat}\text{: N}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet\end{bmatrix} &
\begin{bmatrix}\bullet\textsc{cat}\text{: P}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet\end{bmatrix} &
\begin{bmatrix}\bullet\textsc{cat}\text{: V}\bullet \\ \bullet\textsc{cat}\text{: N}\bullet \\ \textsc{tense}\text{: }\{\textsc{past}, \textsc{pres}\}\end{bmatrix}
\end{array}
$$

- We can then treat the regular morphological endings like *-ed* and *-s* as the result of regular realization rules as in (42).
- This massively reduces the redundancy of listing verb forms like *showed* and *killed* as lexical entries in addition to *show* and *kill*. We express the regularity of this morphology.
- For so-called 'irregular' morphology like *gave* as the past tense form of *give*, we will have to list this as the form realized when *give* combines with a *v* that has a past tense feature value.

(42) *Morphological realization rules*:

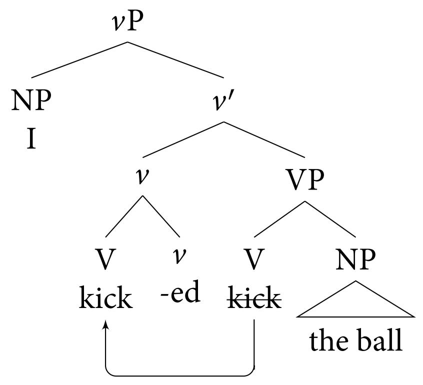
a. $\underset{[\textsc{tense: past}]}{v} \to$ *-ed*

b. $\underset{[\textsc{tense: pres}]}{v} \to$ *-s*/*-Ø*

c. 

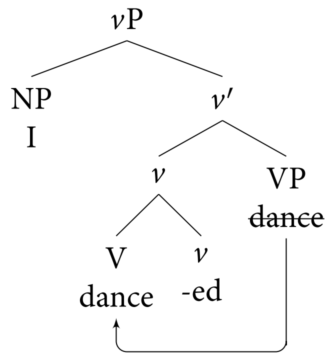
- So, we have now revised our theory to include an additional head *v* (formerly V_2) that selects the subject. We will see more reasons for assuming such a head next week.
- This also means that, by hypothesis, the verb will move to *v* to combine with its morphological ending, even with transitive and intransitive verbs when this has no direct effect on word order.
- A transitive sentence now looks like this (I have omitted selectional features):

(43) I kicked the ball



- An intransitive sentence now looks like this:

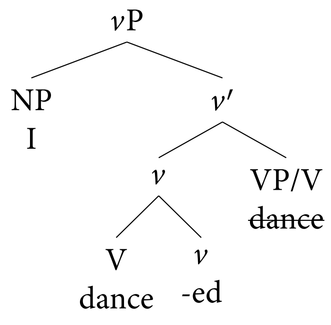
(44) I danced



- There is, of course, something strange going on in (44). Since *dance* does not contain any unchecked features, it counts as a phrase. We have also said that phrases are maximal projections (which they often are). If *dance* is a maximal projection of V, why can it undergo head movement to *v*?

- A common response to this is that, while *dance* is a phrase and a maximal projection of *dance* by some criteria (i.e. regarding (un)checked features), it is also in some sense a minimal projection of *dance* (it is the first and only projection of *dance*). So, in this way it counts both as a minimal and maximal projection. We could represent this as in (45) (but we won't adopt this practice going forward).[3]

(45) I danced



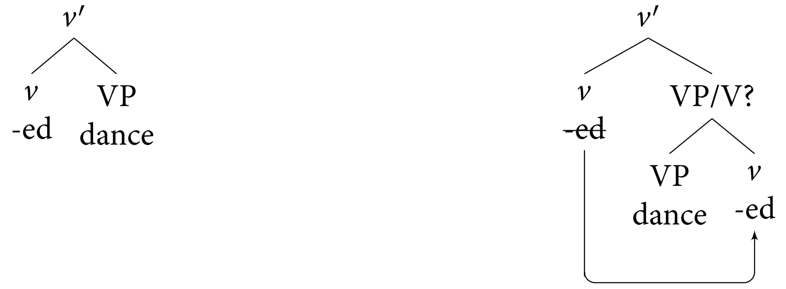
- The definition of head movement, repeated below, stated that the moved element has to be a minimal projection (a necessary but not sufficient condition for being a head), so *dance* can still move to *v*.

Head movement

Create a new node Z with daughters X and Y, where X is a minimal projection (≈head) and Y is the closest c-commanding head to X. Do not pronounce the lower occurrence of X. Also, Z = Y.

- Even when we say this, one might wonder whether it isn't equally possible for *v* (*-ed*) to move to VP *dance* (since the VP) also c-commands *v*:

[3]One might wonder whether the same applies to the NPs in the tree – are these also heads that can undergo head movement? There answer is 'no', at least for English. It will become clear why once we fully spell out what the structure of the English noun phrase is in later weeks.

(46)


- In practice, it might be hard to show that this isn't what happens, but it doesn't fit the general pattern of what we assume is going in the other cases (head movement from V to *v*).
- For now, the definition of head movement precludes this possibility anyway because the target of movement (Y in the definition) is stipulated as being a head, which *dance* in (46) is not (it is a phrase that also counts as a minimal projection).
- Practically, this problem won't arise once we introduce the relevant feature that triggers head movement (that we will place on *v*, not V). But we haven't got to that point yet!

3 Adjuncts and linearization

- So far, we have been making intuitive assumptions about how trees are mapped onto linear order in a sentence (this was reflected in the way we draw them).
- But all these trees encode, i.e. all that Merge creates, is constituency.
- We need some way of deriving linear order from this.
- Below is a first pass at this:

Linearization (version 1)

1. Precedence relations hold between terminal nodes in the tree.
   (This is encoded as X < Y ('X precedes Y').)

2. Intermediate projections are irrelevant for linearization.

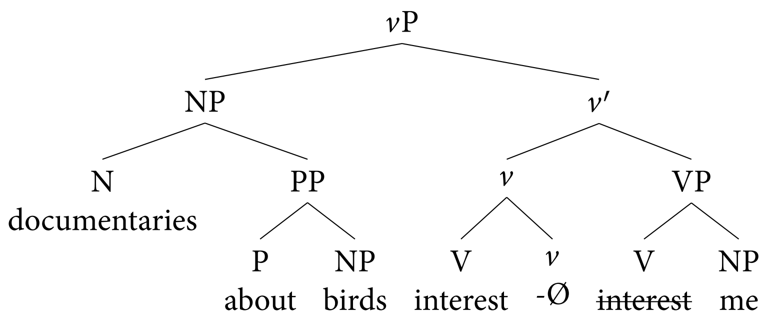
2. A terminal node X precedes another terminal node Y if
   a. X asymmetrically c-commands Y, or
   b. A node Z that dominates X asymmetrically c-commands Y

3. A head either precedes or follows its complement (and everything dominated by its complement).

- Notice the importance of c-command as being the basic relation for deriving precedence.
- For English, we can assume that a head always linearly precedes its complement (i.e. its sister).

- With these assumptions, let's consider how the Linearization procedure above works for the following sentence and its corresponding tree structure:

(47) Documentaries about birds interest me



- Since heads precede their complements in English, we can generate the following linearization statements for the terminal nodes corresponding to the heads N, P and (the higher) *v*. (Let's just assume that the complex *v* head containing *interest* and *v* is linearized like a single terminal).

(48) *Linearization statements (v.1)*

a. documentaries ≺ {about, birds} (N precedes PP)
b. about ≺ birds (P precedes NP)
c. interest-Ø ≺ me (*v* precedes VP)

- Now, since the NP in the specifier of *v* c-commands everything dominated by *v*′, we can add these statements to our list (note that *documenatries* also asymmetrically c-commands *about* and *birds* but this would result in the same statement as (48a)).

(49) *Linearization statements (v.2)*

a. documentaries ≺ {about, birds}
b. about ≺ birds
c. interest-Ø ≺ me
d. {about, documentaries, birds} ≺ interest-Ø (NP asymmetrically c-commands *v*)
e. {birds, documentaries, about} ≺ me (NP asymmetrically c-commands NP)

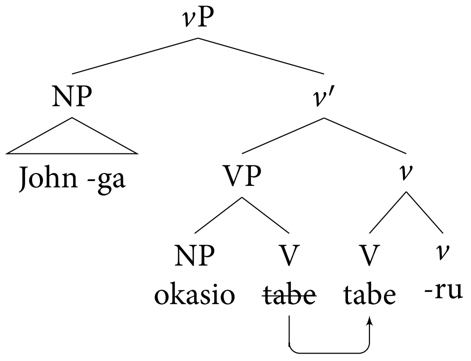
- If we put this all together, the only string compatible with all of these statements is the following:

(50) documentaries ≺ about ≺ birds ≺ interest-Ø ≺ me

- This is the word order we want.

- In other languages, heads follow their complements. So *v*/V will actually follow the object. We can see this with Japanese, for example:

(51) John-ga okasio tabe-ru
John cakes eat-s
'John eats cakes.'



- While the heads are linearized after their complements, the NP/everything contained in the NP that asymmetrically c-commands other terminals in the tree precedes those terminals (e.g. the subject still precedes everything in *v*′).

- Does this mean that nothing can ever be linearized after its complement?

- This seems incorrect for a certain class of phrases that we haven't talked about. These are called **adjuncts**.

- As the name suggests, adjuncts are phrases that can be optionally added to a sentence:

(52) John called me (last night).

- Furthermore, there can be more than one adjunct:

(53) a. John called me (for two hours) (last night)
b. John called me (last night) (for two hours)

- How do these fit with our current views about selection and structure building?

- Well, if we assumed that these are optional arguments of a verb (i.e. posit a selection feature for them), then we run into the problem of lexical redundancy again.

- A transitive verb like *call* would have to be endowed with as many additional selectional features as there are adjuncts.

- It is a reasonable assumption that there is actually not an upper limit on the number of adjuncts we can have (in principle), so this is not a road we want to go down:

(54) John called me (last night) (for two hours) (from his car) (in a huge panic) (all out of breath) …

- Instead, it seems that there must be some mechanism in the grammar that allows us to add phrases as adjuncts freely (i.e. without a selection feature).

- To achieve, let's assume another way of building structures in addition to Merge. We can call this operation **Adjoin**, as defined below.

Adjoin (version 1)

Create a node Z with daughters X and Y, where X is the root node of the current tree and Y is a phrasal constituent in another workspace. If so, then Z = X.

- Note that Adjoin differs from our definition of Merge (repeated below) in that it does not require that a feature is checked as part of this operation.

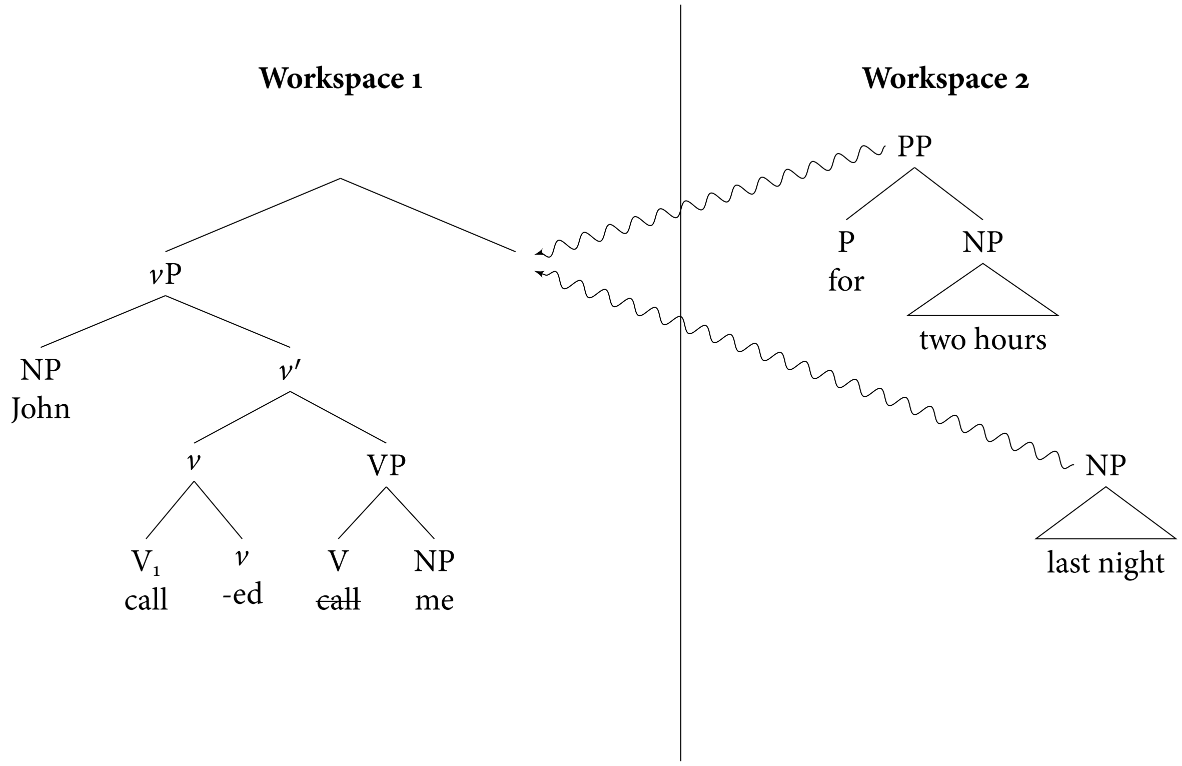
Merge

Create a node Z with daughters X and Y.

Conditions:
a. Only possible if either X or Y has a selection feature ([•CAT: X•] or [•CAT: Y•])
b. If X selects Y, then Z = X (and *vice versa*)
c. If X is a daughter of Y, then do not pronounce X in Y.

- But what does it mean for a phrasal constituent to be 'in another workspace'. A visual representation of two workspaces is given below.

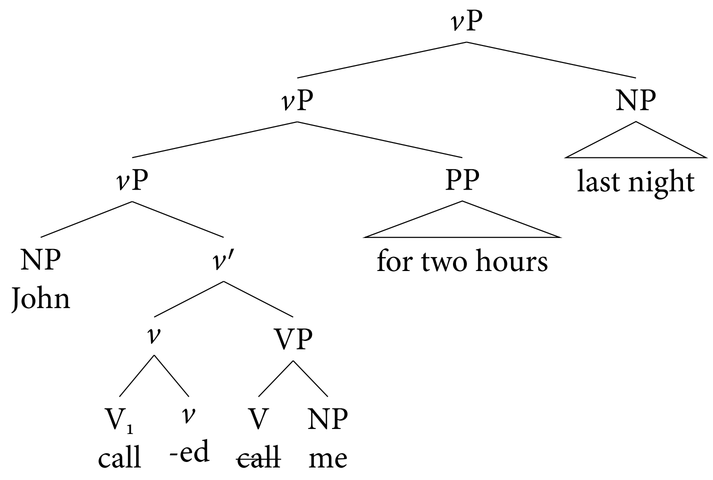
(55)



- Well, think about when we merge the current tree with a syntactic phrase, e.g. *for two hours* or even the complex subject *documentaries about birds* in (47). These are phrases built in the syntax, *for two hours* is not taken directly from the lexicon.

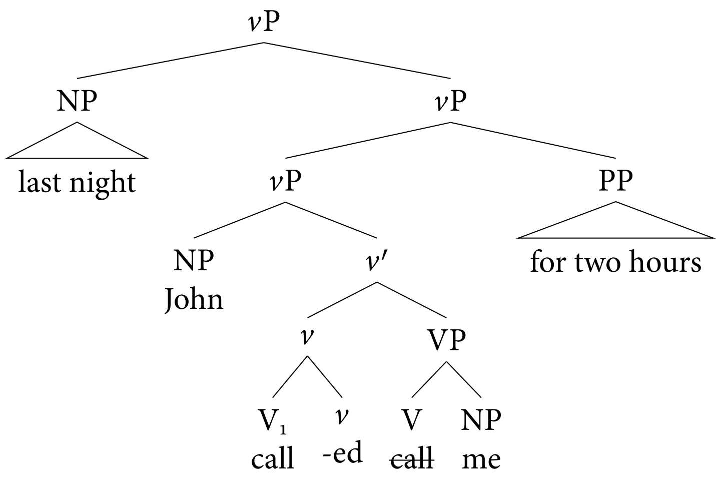
- For this reason, we have to have some other place (we will call it a 'workspace') where we can build phrases to merge into the current tree. That is what Workspace 2 above is. The main tree is built in Workspace 1.
- If a phrase comes from another workspace and this merge does not check a feature, then this meets the definition of Adjoin. The workspace-external phrase is called an 'adjunct' and the resulting node takes the category of one that was adjoined to (it is another maximal projection).
- So merging both of the phrases in Workspace 2 in one of the possible orders gives us the following tree:

(56)



- Linearly, both of the adjuncts in (56) *follow* every terminal node they c-command rather than preceding them.
- Furthermore, it is possible for adjuncts (with some restrictions) to precede everything they c-command, too, as *last night* in (57).

(57)



- In light of this, it seems we need to adapt our linearization to incorporate this.
- Let's therefore add another clause about adjuncts specifically. We want them to be able to either precede

or follow everything they c-command. This can be referred to as **right-adjunction** or **left-adjunction**:

Linearization (version 2)

1. Precedence relations hold between terminal nodes in the tree.
   (This is encoded as X ≺ Y ('X precedes Y').)

2. Intermediate projections are irrelevant for linearization.

2. A terminal node X precedes another terminal node Y if
   a. X asymmetrically c-commands Y, or
   b. A node Z that dominates X asymmetrically c-commands Y

3. A head either precedes or follows its complement (and everything dominated by its complement).

4. An adjoined node (and everything dominated by this node) either precedes or follows every node that it c-commands.

- This will get us the right result for (57): the left-adjoined NP *last night* c-commands, and therefore precedes, all terminal nodes corresponding to *John called me for two hours*, while right-adjoined PP *for two hours* c-commands, and therefore follows, all terminal nodes corresponding to *John called me*.

- It is worth noting, however, that adjuncts can't adjoin everywhere. We need to have some restrictions. Consider the data in (58).

(58) a. I destroyed the vase quickly.
 b. I quickly destroyed the vase.
 c. *I destroyed quickly the vase.

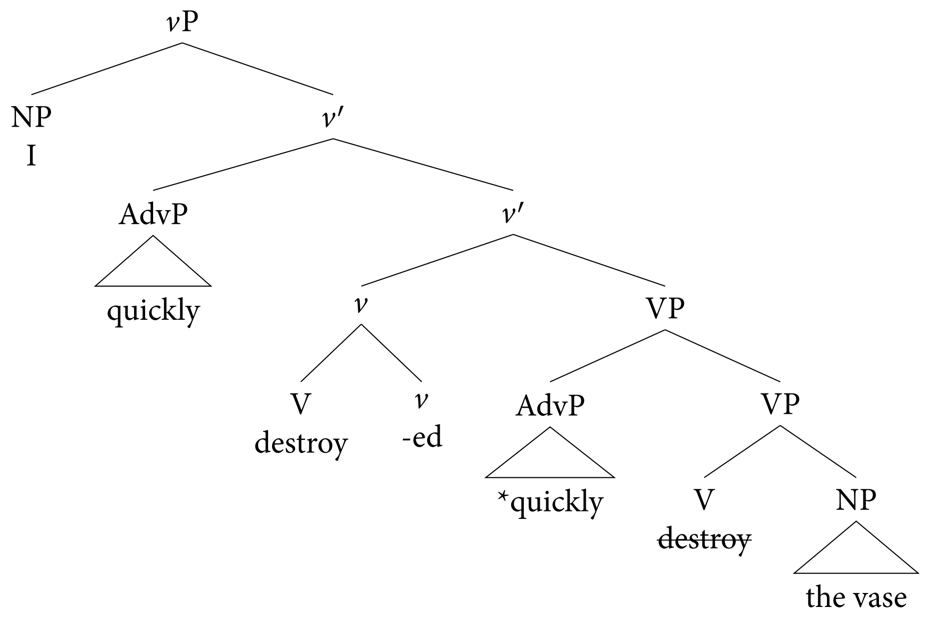
- Right-adjunction of *quickly* is possible. We can assume this is adjunction to *v*P, as above.

- We see that *quickly* can surface between the subject and verb, but not between the verb and object. One possibility is that the sites for left-adjunction are more restricted.

- We could assume, for example, that *quickly* may left-adjoin to *v*′, but not to VP. This would appear to get the kind of restriction we want to rule out (58c), as shown in (59).

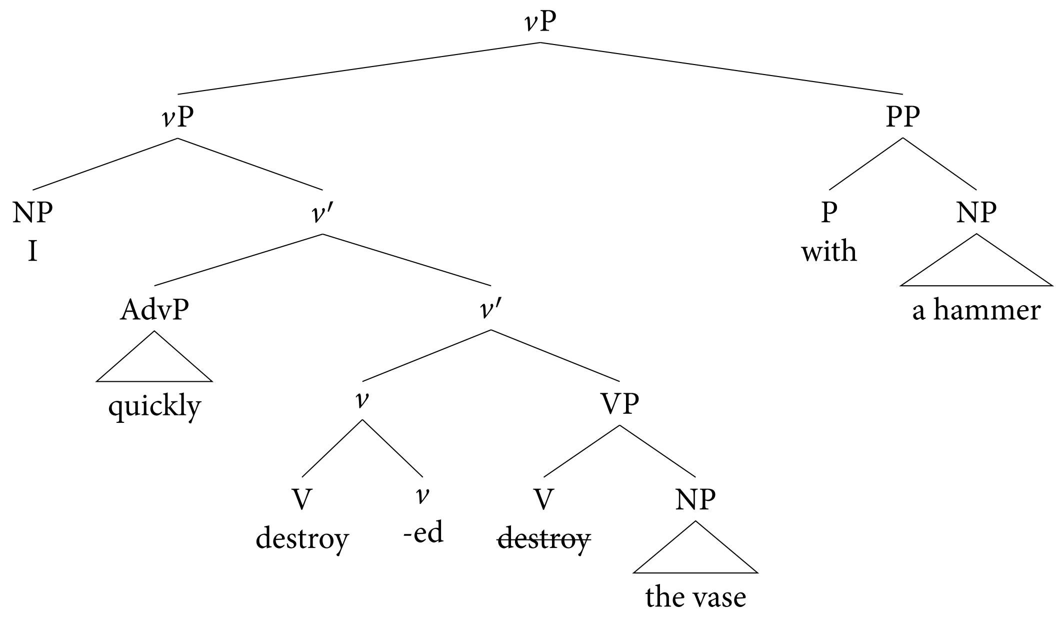
- (Note that I am treating *quickly* as of category Adv. It seems plausible that the head of this phrase is *-ly*, selecting an adjective as its complement. An issue here is that the head would have to follow its complement, which is unusual. It is possible, however, that the property of a head being linearized before or after its complement is a property of a given lexical item (head) and may vary even within a language.)

(59)

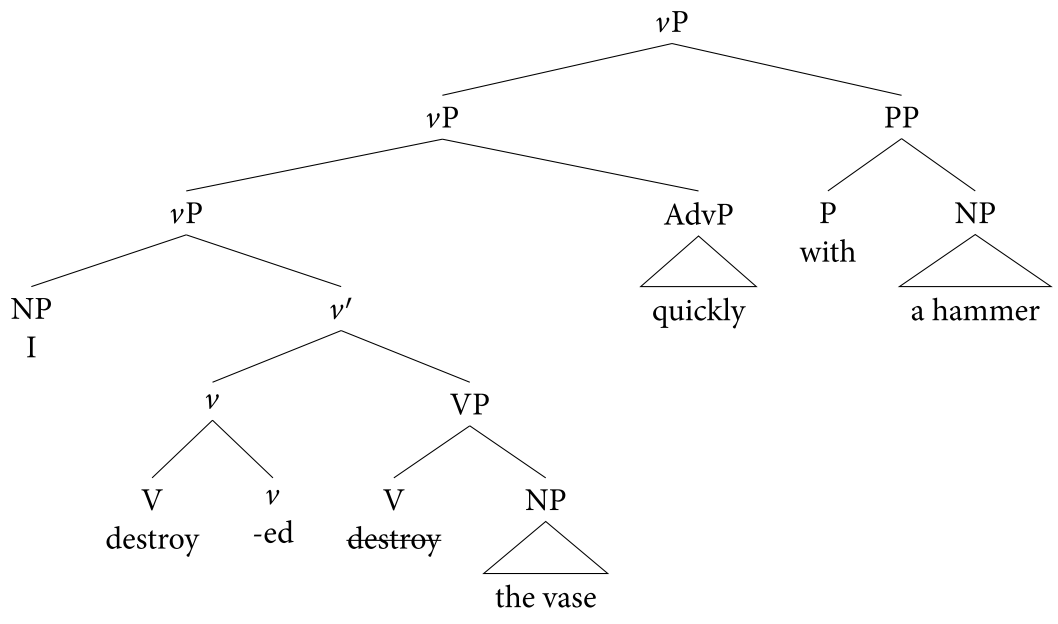


- This gives us various different options: *quickly* may adjoin either to the left or right but ,for some reason, *with a hammer* may only adjoin to the right.

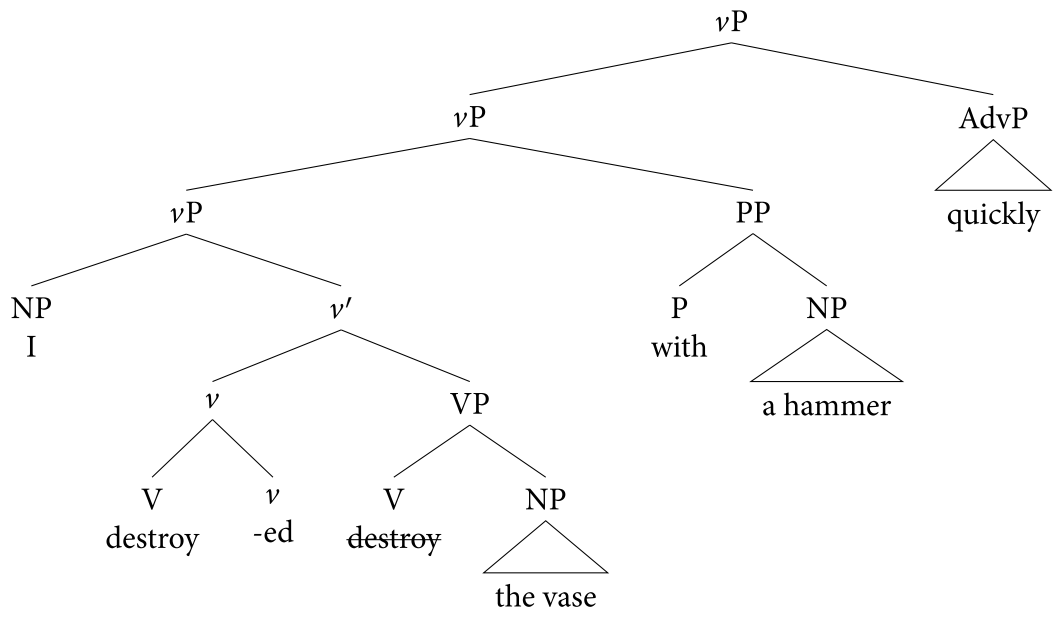
(60) I quickly destroyed the vase with a hammer.



(61) I destroyed the vase quickly with a hammer



(62) I destroyed the vase with a hammer quickly



- Note that it is possible that the right-adjoining adjuncts are attaching lower. In practice is quite hard to know. In general, I will adjoin them as high as possible, i.e. with the regard to what gives the correct linearization.

- The final thing I want to discuss is the following: Is it possible that we can also adjoin something that is already in the tree?

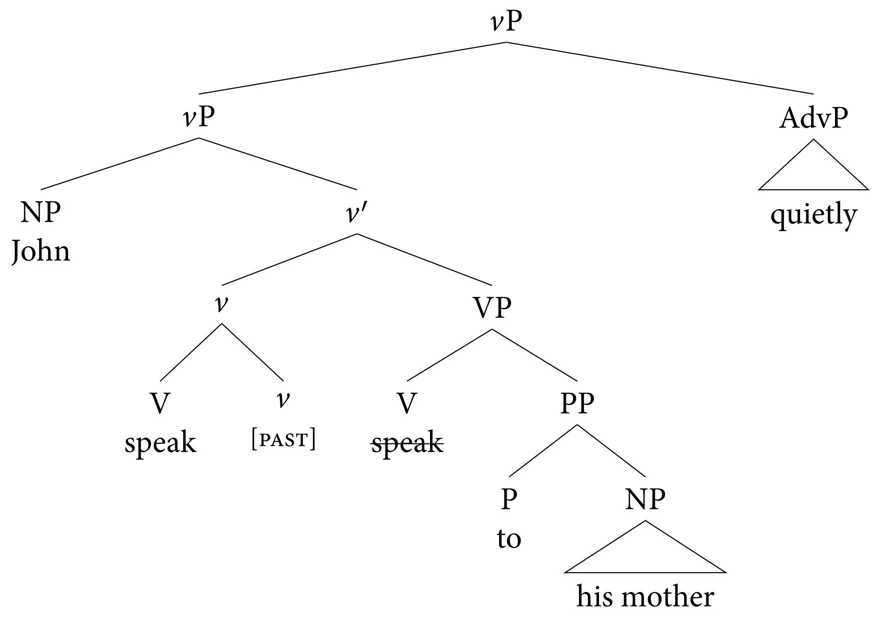
- The prediction would be that this moved phrase could potentially end up following everything it c-commands (rather than preceding it like most moved things do).

- Indeed, examples like the following suggest this:

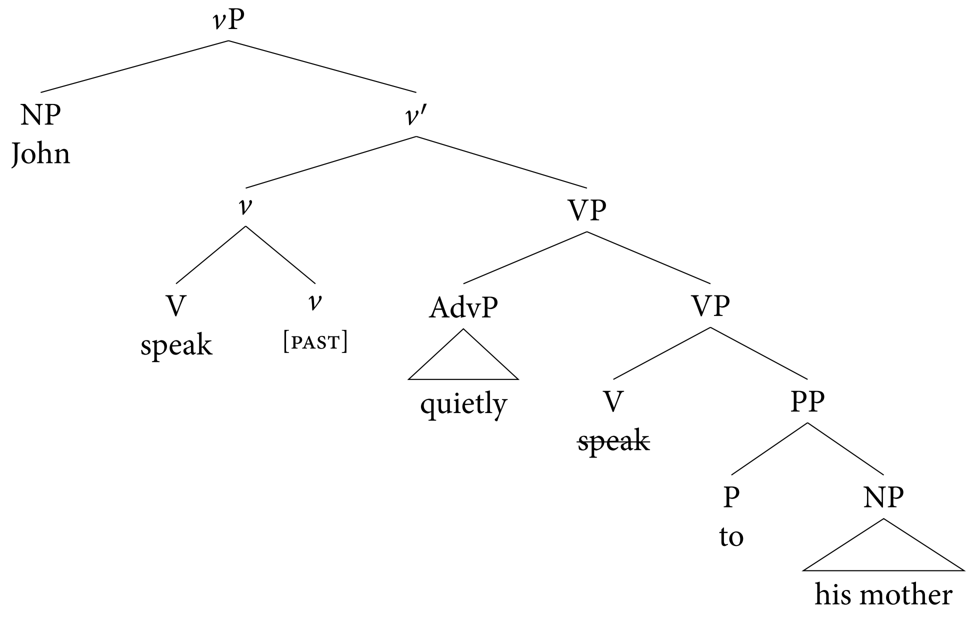
(63) a. John spoke to his mother quietly
 b. John spoke quietly to his mother

- We could have the two following analyses for these sentences:

(64)



(65)

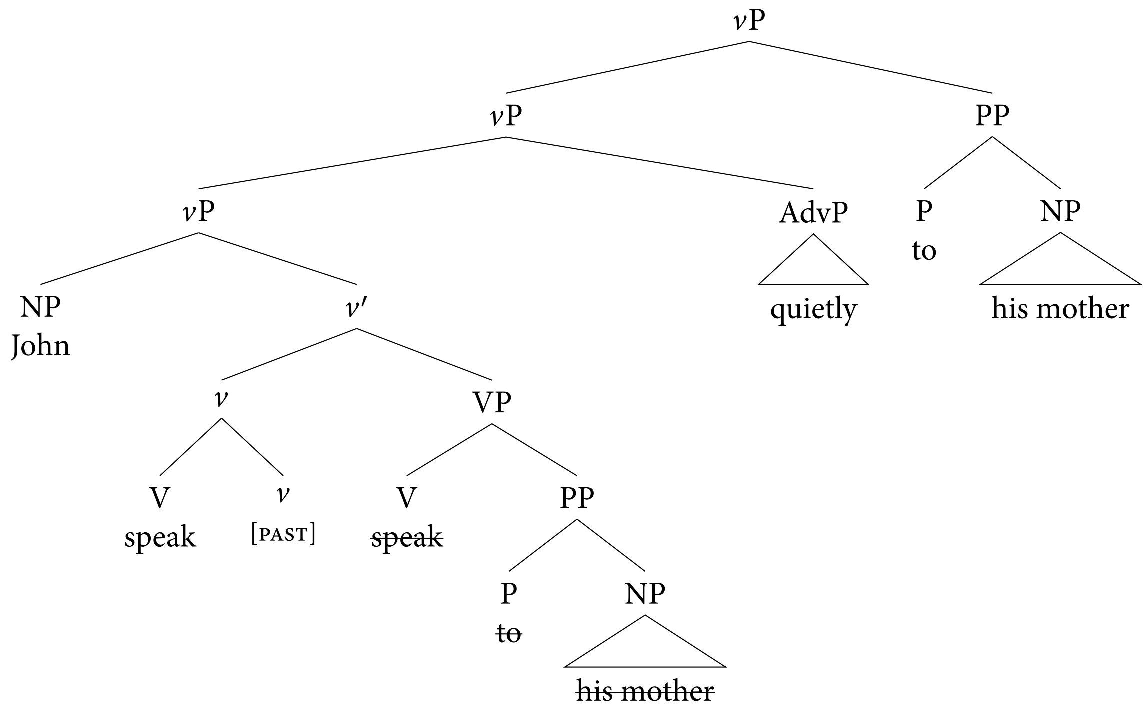


- But what is the problem with (65)? Well, before we showed that left-adjunction to VP did not seem to be possible. Why is it an option here?

- It could be that there is something special about VPs containing a PP object, but ideally that isn't something we would want to say. The other option is that what is actually going on here is that the PP is being displaced.

- This displacement is different, however. If it is moving higher in the tree, it must land in a position where it follows everything it c-commands. As we have seen, this an adjoined position. The structure would therefore look as follows:

(66)



- This kind of movement landing in a right-adjoined position is sometimes called **extraposition**.

- In light of this possibility, we should adapt our definition of Adjoin to allow for adjunction under displacement. In English, we will assume that only right-adjunction is possible for option (b).

Adjoin (version 2)

Create a node Z with daughters X and Y ('adjoin Y to X').

Conditions:
a. If X is the root node of the current tree and Y is a constituent in another workspace, then Z = X.
b. If X is a daughter of Y, then Z=Y and do not pronounce X if dominated by Y.

- Is this option only possible to PP complements of the verb? Perhaps not. It has been noticed that the apparently ungrammatical placement of *quickly* in (67a) (repeated from above) becomes more acceptable when the NP in question is more complex (67b).

(67) a. *He destroyed in only ten seconds the vase.
b. He destroyed in only ten seconds the most expensive vase that I had ever laid eyes on.

- We could assume that a noun phrase can undergo the kind of movement we saw in (66) only if it meets this 'complexity' requirement.