CAS LX 321 / MET LX 521 / GRS LX 621 *Syntax: Introduction to Sentential Structure*

March 1, 2024

1 Subject agreement in the age of DP

1.1 Getting agreement and tense to the verb

Now that we've clarified the structure of DP, let's think a bit about how the features are supposed to work to get subject agreement.

The inflection on a verb correlates with properties both of the subject and of tense. This is what we are trying to account for.

(1) a. They see us.
    b. You see us.
    c. It sees us.
    d. They saw us.
    e. You saw us.
    f. It saw us.

We want our grammar to account for this, to rule out cases of mismatch. This could be done by verifying that we pick matching subjects, tenses, and verb forms, under the assumption that a feature clash (like [+pl] and [–pl] together) results in an ungrammatical sentence. Or we could say that the verb doesn't even have features until it gets them from tense and the subject (so mismatch is not even possible). This latter view is the one we'll take.

Under this view, there are two things we know about a verb like *see*, two things that are part of the lexicon. We know its syntactic properties (2), we know its morphophonological properties (3), and we have a "redundancy rule" that characterizes V generally (so we don't need to mark it in every lexical item) (4).

(2) *see*, V, [+ _ DP]

(3) a. [–speaker, –addressee, +sg, –pl, –past] → *sees* 3sg present
    b. [+past] → *saw* past
    c. [ ] → *see* otherwise

(4) V << [?T-ϕ] V always has [?T-ϕ] (needs tense features and *phi*-features)

We assume that tense is the "home" of past/present and infinitive/finite information. That the difference between a present tense sentence and a past tense sentence is just in the choice of T. So, let's suppose we have three lexical items corresponding to the different types of T.

(5) a. ∅, T, [+finite, +past] past tense
    b. ∅, T, [+finite, –past] present tense
    c. ∅, T, [–finite] infinitive

We need to get both tense and subject agreement information down to the verb, because the verb has a [?T-ϕ] feature (the "?" indicates a blank, a missing feature, something to fill). The subject is in the specifier of TP, so we assume that the subject shares its features with T, and then T (which now has the subject's features and its own) shares its features with V. So we assume that T always has a [?ϕ] feature (needs *phi*-features, which it will gather from the subject).

(6) T << [?ϕ] T always has [?ϕ]

We assume that the ϕ-features are a property of a DP. So something like *the squirrel* is third person singular ([–spkr, –addr, +sg, –pl]) and *the squirrels* is third person plural ([–spkr, –addr, –sg, +pl]). These features need to get to T to give a value to its [?ϕ] feature. Since subject-verb agreement is truly something about subjects, and subjects are in the specifier of TP, we hypothesized a **Spec-head agreement** rule.

Spec-Head Agreement: Features can be shared between the specifier of a phrase and its head.

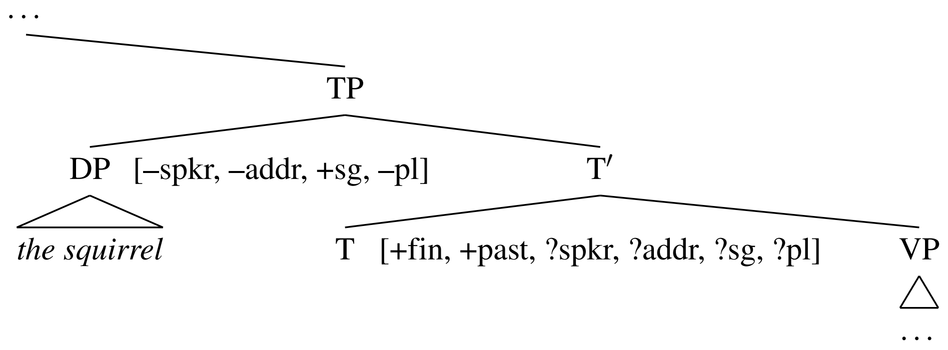
This is a basic pathway, but what features are shared? Is the travel one-directional (just from specifier to head)? Or can features of the head also be shared with the specifier? So far, we only need ϕ-features to be transferred from the specifier to the head. But thinking ahead a bit, we will want to say that T can share features with the subject in its specifier in order to establish nominative case on the subject. So this seems to be a two-directional pathway. We will return to case, let's focus on the ϕ-features first. For the moment, let's assume that we have a list of features that are shared between specifier and head (so not all of them are shared, just the ones in our list). Our list currently contains ϕ-features (person, number, gender features). Another possibility is that the only features that get shared are those that fill in an unvalued ("?") feature.

For English (which doesn't realize gender features), this probably amounts to something like this:

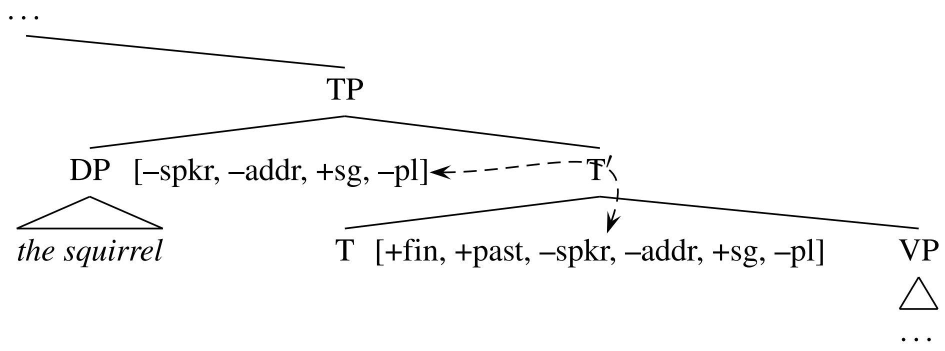
[+speaker, –addressee]	1st person
[–speaker, +addressee]	2nd person
[–speaker, –addressee]	3rd person
[+speaker, +addressee]	2nd person inclusive?
[+sg, –pl]	singular
[–sg, +pl]	plural
[–sg, –pl]	mass

And then we'll assume that [?ϕ] (the feature that we have on T) is short for [?speaker, ?addressee, ?sg, ?pl]. On T, these start off as "unset" or "unvalued" but when the features are shared between specifier and head, the unvalued ones are resolved to have the values that match the valued ones.

So we start off with something like this:


. . .
TP
DP [–spkr, –addr, +sg, –pl]
T′
the squirrel
T [+fin, +past, ?spkr, ?addr, ?sg, ?pl]
VP
. . .

And after Spec-head agreement, wind up with this:

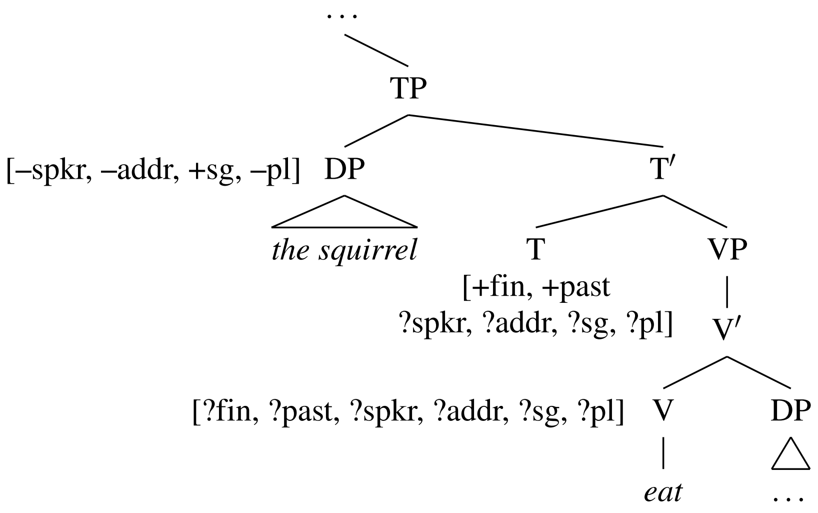

...
TP
DP [–spkr, –addr, +sg, –pl]
T′
the squirrel
T [+fin, +past, –spkr, –addr, +sg, –pl]
VP
...

The next step is to get the features of T down to the verb. We can accomplish this by supposing that T shares features with its complement VP, in combination with supposing that a head and its phrase share features. So T shares features with VP which shares features with V. We know that feature percolation, the sharing of features between VP and V, needs to be bidirectional (since passing features to VP sends them down to V, but features of P are sent up to PP for selection). We don't know that VP inflection needs to be bidirectional, it was stated in a one-directional (head to complement) way.

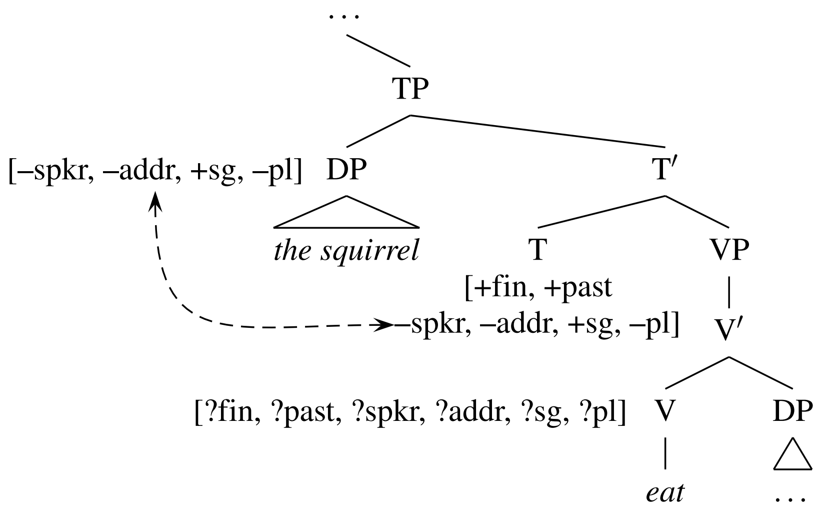
VP inflection: T passes (certain) features to a sister VP.

Feature percolation: (Certain) features are shared between a phrase and its head.

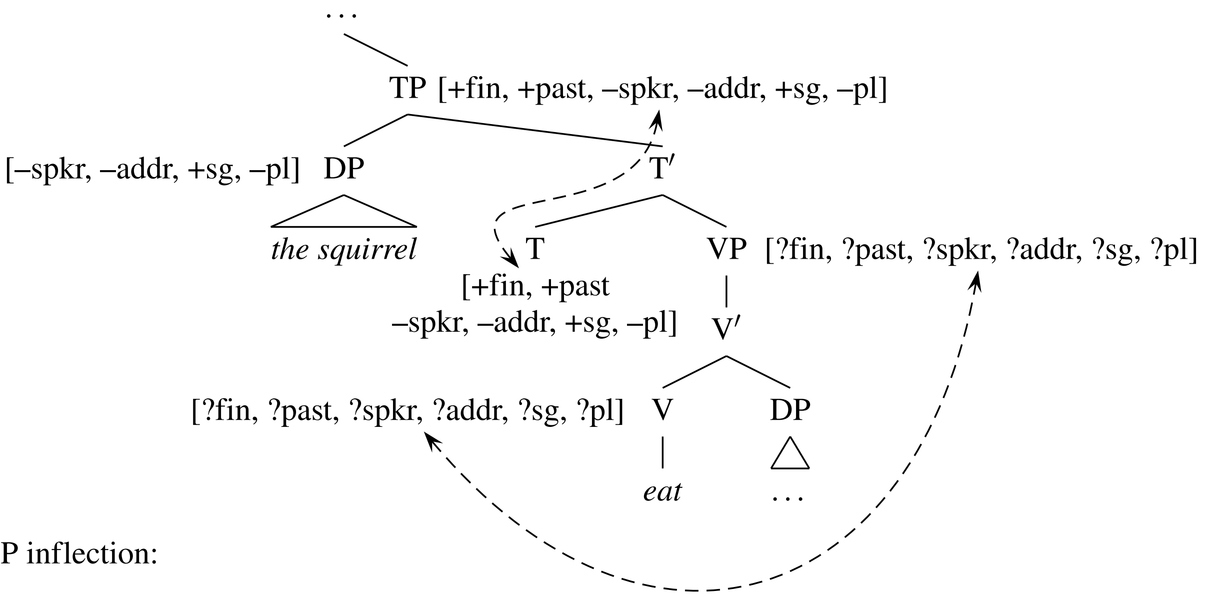
So continuing on in the squirrel tree, we would start off with:



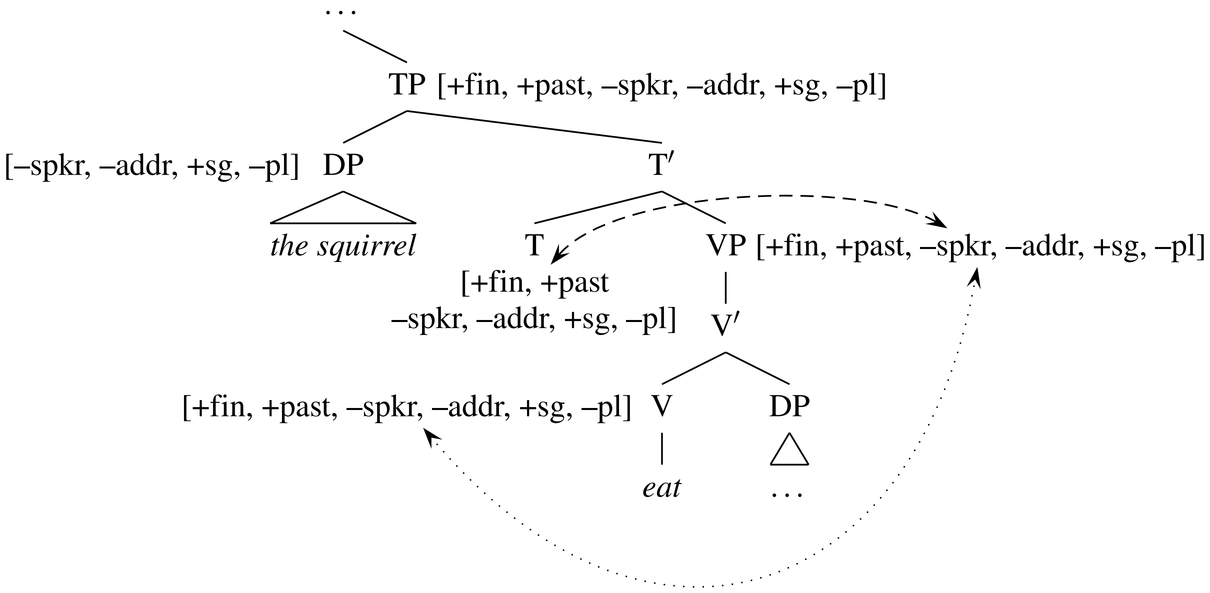
Spec-head agreement:



Percolation:


. . .
TP [+fin, +past, –spkr, –addr, +sg, –pl]
[–spkr, –addr, +sg, –pl] DP
T′
the squirrel
T
[+fin, +past
–spkr, –addr, +sg, –pl]
VP [?fin, ?past, ?spkr, ?addr, ?sg, ?pl]
V′
[?fin, ?past, ?spkr, ?addr, ?sg, ?pl] V
DP
eat
. . .

VP inflection:


. . .
TP [+fin, +past, –spkr, –addr, +sg, –pl]
[–spkr, –addr, +sg, –pl] DP
T′
the squirrel
T
VP [+fin, +past, –spkr, –addr, +sg, –pl]
[+fin, +past
–spkr, –addr, +sg, –pl]
V′
[+fin, +past, –spkr, –addr, +sg, –pl] V
DP
eat
. . .

And then the way *eat* is pronounced when it is finite, past, third person (neither addressee nor speaker), singular: *ate*.

For the sake of typesetting efficiency, we can use shorthand for several of these combinations. (As for why there don't seem to be realizations of "1mass" or "3dual," we'll leave that for some other time.)

[+speaker, –addressee, –sg, +pl]	[+1pl]
[+speaker, –addressee, +sg, –pl]	[+1sg]
[–speaker, +addressee, –sg, +pl]	[+2pl]
[–speaker, +addressee, +sg, –pl]	[+2sg]
[–speaker, –addressee, –sg, +pl]	[+3pl]
[–speaker, –addressee, +sg, –pl]	[+3sg]
[–speaker, –addressee, –sg, –pl]	[+3mass]
[+finite, –past, –speaker, –addressee, +sg, –pl]	[+pres3sg]
[+finite, +past, –speaker, –addressee, +sg, –pl]	[+past3sg]
etc.	

1.2 Getting ϕ-features to DP

Let's now think about *the squirrel.* That was labeled as being [+3sg], but given that it has internal structure, how does that arise?

Let's suppose that [+3sg] is a property of the head D (and that it comes to be a property of DP through percolation), and suppose that N gets its number feature from D (in much the same way that V gets its tense features and ϕ-features from T). So N is inflected in much the same way as V is. If the NP is a complement of a plural D, we get a plural pronunciation and if it is a complement of a singular D, we get a singular pronunciation.

(7) *squirrel*, N

(8) a. [–pl,] $\rightarrow$ *squirrel* — singular/mass

b. [ ] $\rightarrow$ *squirrels* — otherwise

(9) N << [?sg, ?pl] — N always has [?sg, ?pl] (needs a number specification)

Then we have determiners like these:

(10) a. the, D, [+3sg] — singular definite determiner

b. the, D, [+3pl] — plural definite determiner

c. this, D, [+3sg] — singular proximal determiner

d. these, D, [+3pl] — plural proximal determiner

e. that, D, [+3sg] — singular distal determiner

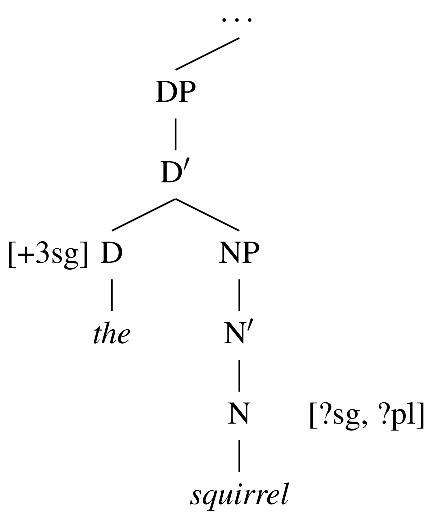
f. those, D, [+3pl] — plural distal determiner

g. a, D, [+3sg] — singular indefinite determiner

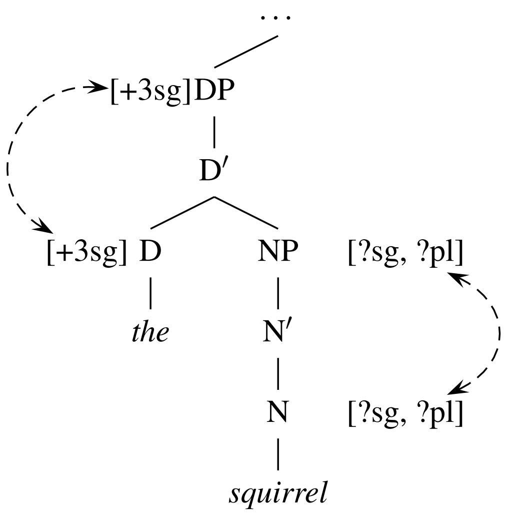
h. ∅, D, [+3pl] — plural indefinite determiner

NP inflection: D passes (certain) features to a sister NP.

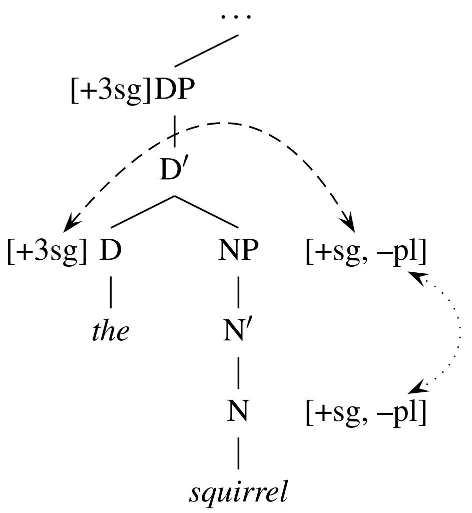
Given that, we get [+3sg] *the squirrel* like this:



Feature percolation:



NP inflection:



1.3 Pronouns and case

We are supposing that pronouns are simple DPs, a DP with no NP inside. They have the ϕ-features inherently (and this is fine with respect to the story above, since that's where ϕ-features were even for *the squirrel*). But one other thing that is clear about pronouns in English is that they show a case distinction.

If we want to be sure that *they* winds up in subject position and *them* winds up in object position, we could check for clashing features somewhere, or we could do the same kind of feature "assignment" that we were doing for subject agreement. For the sake of consistency, we will go with the latter.

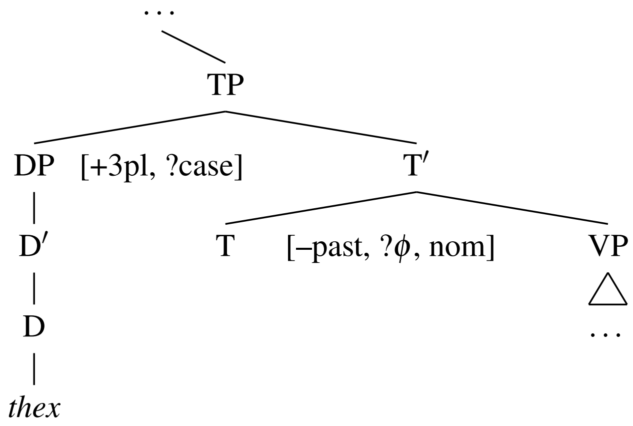
What we will suppose is that DPs (pronouns, but actually any DP) has a feature that corresponds to its case form, which gets its value from somewhere else (like T). What that means is that *them* and *they* are actually the same pronoun, just pronounced differently depending on what value they get for the case feature (which corresponds to where they are in the structure). Like how *squirrel* and *squirrels* are the same noun, just pronounced differenly depending on whether they are in a plural DP or not.

We may well be able to break down case forms into component binary features (possibly different numbers of them depending on the language), but let's set that aside for now and just suppose that there are features like [nom], [acc], [gen] (for nominative, accusative, genitive) that correspond to cases, and that if a D has [?case] (a case feature without a value) then any of those can give it a value. The basic example is the subject getting nominative case from T. We will suppose that T has a [nom] feature, and D has a [?case] feature.

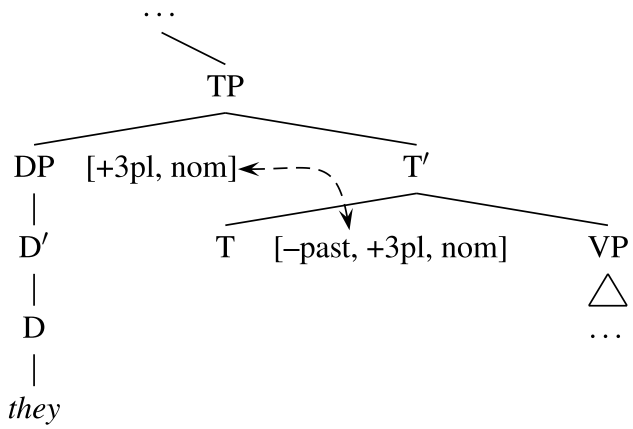
(11) D << [?case] D always has [?case] (needs a case feature)

(12) T [+fin] << [nom] Finite T always has a [nom] feature

We start with this:



And then spec-head agreement yields:



As for accusative case, we get accusative case at least in the objects of prepositions and verbs. We saw in a homework that English will also use an accusative case form when a pronoun is left without case, but let's set that aside for now and define V and P as having the ability to provide an [acc] case value to a DP.

(13) P [+_ DP (...)] << [acc] A P with an object always has an [acc] feature

(14) V [+_ DP (...)] << [acc] A V with an object always has an [acc] feature

This probably isn't quite right, but it's close-ish.

To make this work, we need yet another couple of rules transferring features between a head and its complement. We already had the VP inflection rule (features from T to VP) and the NP inflection rule (features from D to NP), and now we want to add V to DP, and P to DP.

V-DP inflection: V passes (certain) features to a sister DP.

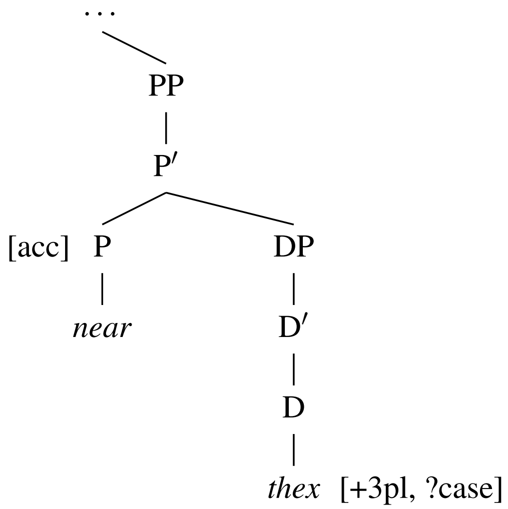
P-DP inflection: P passes (certain) features to a sister DP.

At this point, we seem to have collected too many of these "head passes (certain) features to its complement" rules. This seems like a general relation, like specifier-head. We don't want to allow all features to cross this boundary, since the result of that would just be that every feature gets everywhere. We've seen this feature transfer for ϕ-features, tense features, case features. But in particular, we've seen it *when there is a need for a feature*. So, perhaps we can get away with generalizing it as allowed when it is solving a problem. A head and complement can value each other's features. And in fact, that's what we've seen with head and specifier too. So perhaps we can generalize all of those to:

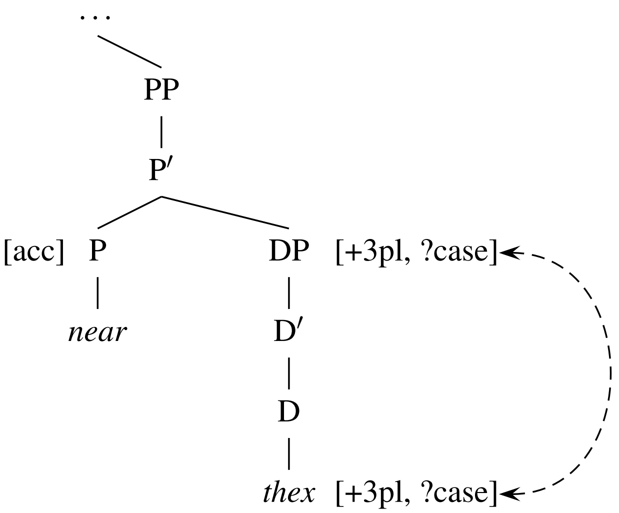
Local agreement: An unvalued feature can get a value from another if it is close.

Close: The specifier and complement are close to the head.

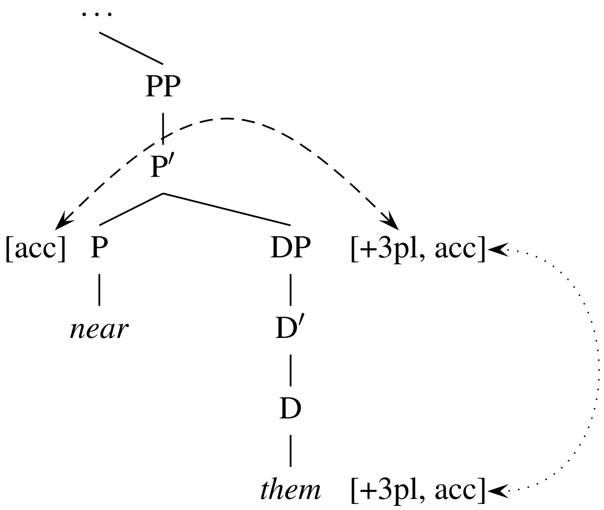
Now, let's look at accusative case in the PP.



Percolation:



Local agreement:



1.4 Summary

In summary, we have the following redundancy rules:

(15) V << [?T-ϕ] V always has [?T-ϕ] (needs tense features and *phi*-features)

(16) N << [?sg] N always has [?sg] (needs a number feature)

(17) D << [?case] D always has [?case] (needs a case feature)

(18) T [+fin] << [nom] Finite T always has a [nom] feature

(19) P [+_ DP (. . .)] << [acc] A P with an object always has an [acc] feature

(20) V [+_ DP (. . .)] << [acc] A V with an object always has an [acc] feature

And the following rules guiding features around the structure:

(21) **Feature percolation**: (Certain) features are shared between a phrase and its head.

(22) **Local agreement**: An unvalued feature can get a value from another if it is close.

(23) **Close**: The specifier and complement are close to the head.