Pat must not have been sending flowers to Chris (as of the midterm)

Suppose that you were faced with the task of generating a structure for *Pat must not have been sending flowers to Chris*.
Note: I will give you a revised version of this about a week after the midterm with some additions.

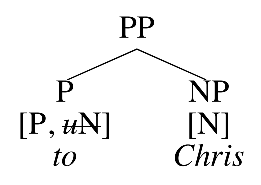
First, assemble your numeration. What are the elements in this sentence?

T	[T, *u*N, past]	There is always a T in a sentence. T always (in English) has [*u*N]. (This really only got a passing mention in class so far, but I include it here for completeness.) *Must* is past tense (cf. *may*), so we know there is [past]. (This too was only mentioned quickly in class, and we'll return to it. But *must* is the past form of *may*.)
v	[*v*, *u*N]	There is always a *v* in a sentence. *give* is agentive, so this *v* assigns the Agent θ-role; hence it has a [*u*N] feature.
Not	[Neg]	*not* is of category Neg.
must	[M]	We have the modal *must*, which is of category M.
Perf	[Perf]	We have perfective *have*, which is of category Perf.
Prog	[Prog]	We have progressive *be*, which is of category Prog.
send	[V, *u*N, *u*P]	*send* is a ditransitive (that is, between V and *v* there are three θ-roles to give out. *v* is responsible for Agent, and V is responsible for the other two. Hence, we have [*u*P] corresponding to the Goal θ-role and [*u*N] corresponding to the Theme θ-role).
to	[P, *u*N]	We have the preposition *to*. Prepositions always have [*u*N] for their object.
Chris	[N]	
Pat	[N]	
flowers	[N]	

Then, start at the bottom (generally the right edge of the sentence in an SVO language like English), and work your way up.

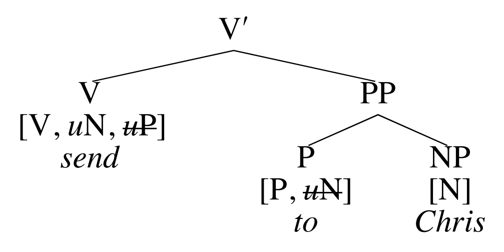
Step 1. Merge *to* and *Chris*.

This puts the [*u*N] feature of *to* in close proximity to the [N] of *Chris*: [*u*N] is checked.
Since *to* had its feature checked (it motivated the Merge), the features of *to* project.
Since *Chris* does not project further and had no uninterpretable features to check, it is a maximal projection (NP).
Since there are no further uninterpretable features on *to*, the object we just formed is a maximal projection (PP).



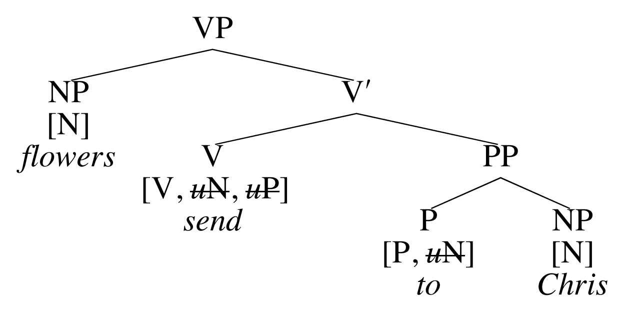
Step 2. Merge *send* and the PP.

This puts the [*u*P] feature of *send* in close proximity to the [P] of the PP. [*u*P] is checked.
Since *send* had its feature checked, the features of *give* project.
Since there is still an uninterpretable feature on *send*, this is an intermediate projection (V′).
The UTAH tells us that *to Chris* is the Goal (PP daughter of V′).
I will put verbs and auxiliaries in their uninflected form (*send*, *be*, *have*) until the last step.



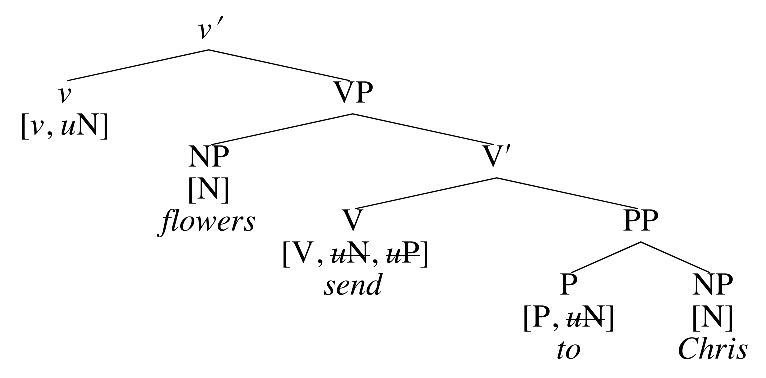
Step 3. Merge *flowers* and the V′.

This puts the remaining [*u*N] feature of *send* close to the [N] feature of *flowers*. [*u*N] is checked.
Since *give* had its feature checked, the features of *send* project.
Since there are no more uninterpretable features on *send*, this is a maximal projection (VP).
Since *flowers* does not project further and had no uninterpretable features to check, it is a maximal projection (NP).
UTAH tells us that *flowers* is the Theme (NP daughter of VP).



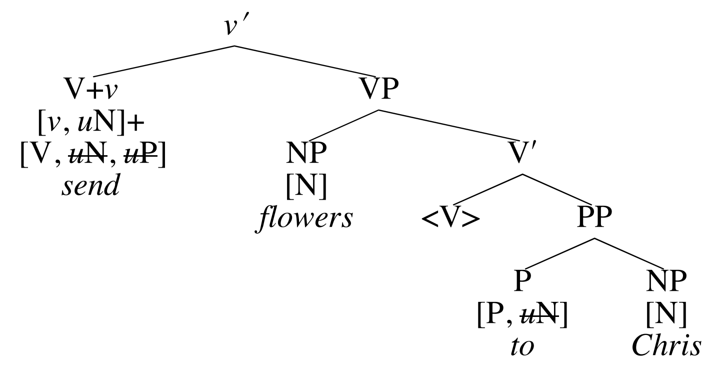
Step 4. Merge *v* and VP.

We are finished with V now, and *v* is the next step up on the Hierarchy of Projections.
No features are checked in this step.
Because *v* is higher than V on the hierarchy of projections, the features of *v* project.
Since *v* still has an uninterpretable feature to check, this is an intermediate projection (*v*′).



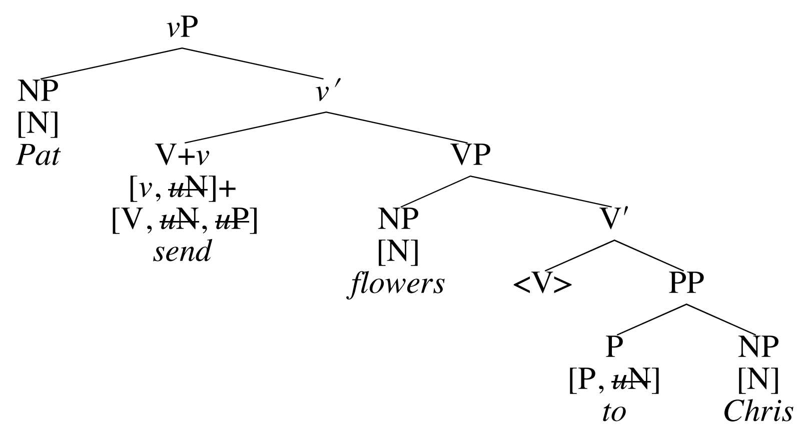
Step 5. Move V to *v*.

V always moves to *v*. Soon we will have a feature that causes this, for now it is just a stipulation.
We show this by replacing *v* with V+*v* and putting the original position of V in brackets: <V>.
I will list both sets of features in this "complex head" V+*v*, with the features of *v* on top, and the features of *v* below.



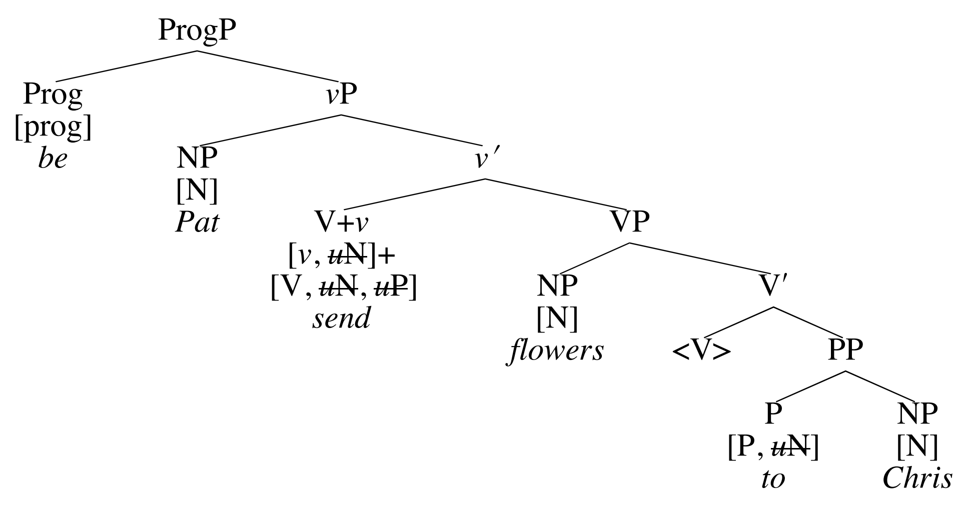
Step 6. Merge *Pat* and *v*′.

This puts the remaining [*u*N] feature of *v* close to the [N] feature of *Pat*. [*u*N] is checked.
Since *v* had its feature checked, the features of *v* project.
Since there are no more uninterpretable features on *v*, this is a maximal projection (*v*P).
Since *Pat* does not project further and had no strong features to check, it is a maximal projection (NP).
UTAH tells us that *Pat* is the Agent (NP daughter of *v*P).
Sometimes NP daughter of *v*P is actually an Experiencer—if it is, we assume that it is a slightly different *v* ($v_{\text{EXPERIENCER}}$) for which NP daughter of *v*P is an Experiencer. But with *send* here, we have v_{AGENT}, not $v_{\text{EXPERIENCER}}$.



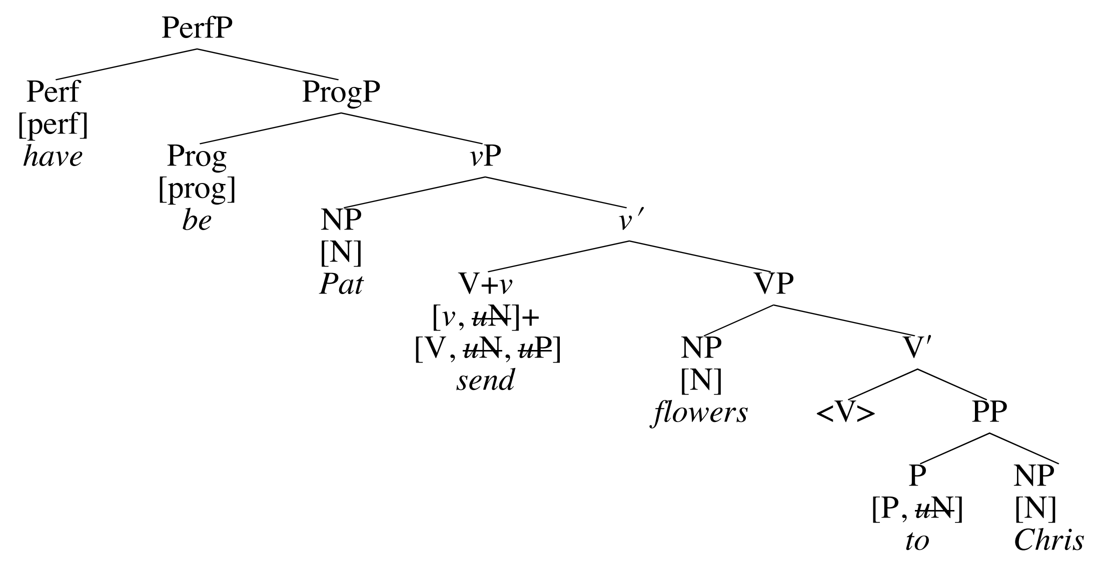
Step 7. Merge Prog and *v*P.

We are finished with *v* now, and Prog is the next step up on the Hierarchy of Projections.
Because Prog is higher than *v* on the hierarchy of projections, features of Prog project.
Since Prog has no uninterpretable features to check, this is a maximal projection (ProgP).



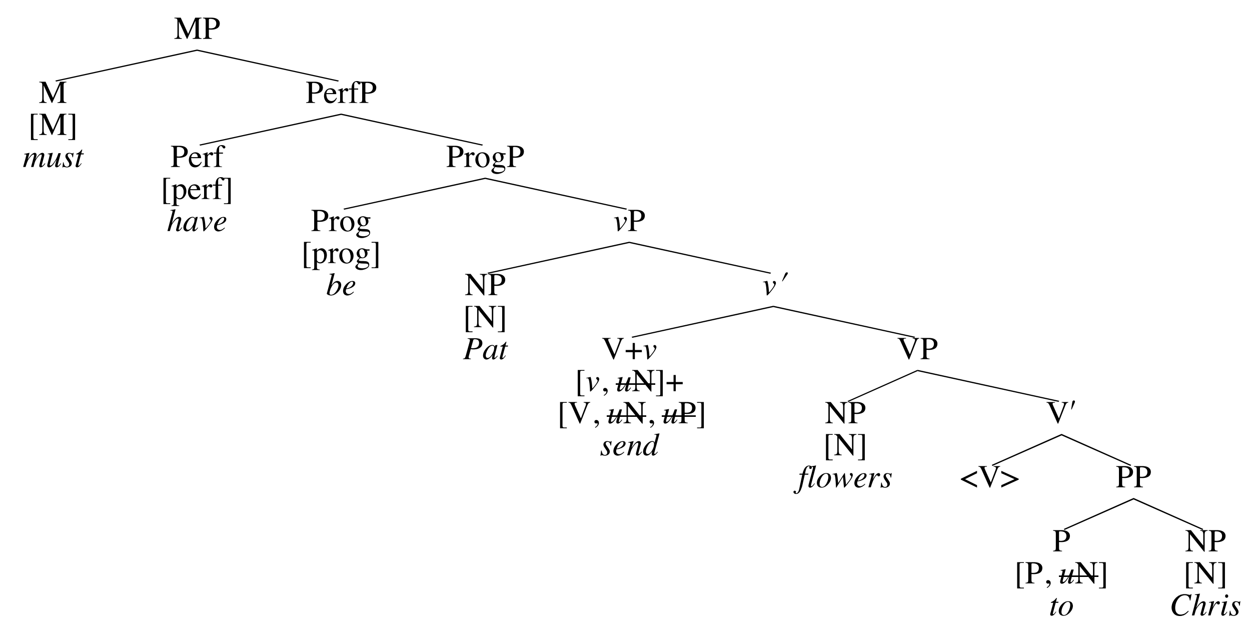
Step 8. Merge Perf and ProgP.

We are finished with Prog now, and Perf is the next step up on the HoP.
Because Perf is higher than Prog on the hierarchy of projections, features of Perf project.
Since Perf has no uninterpretable features to check, this is a maximal projection (PerfP).



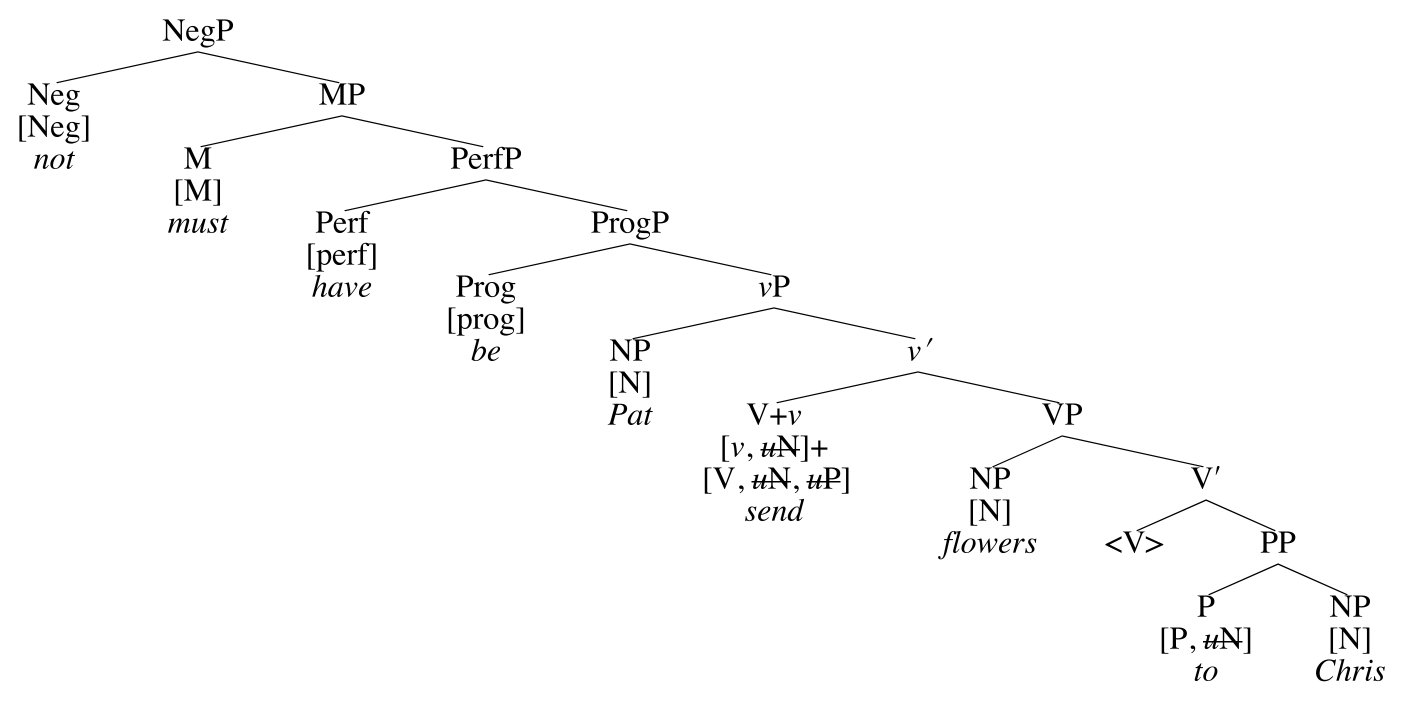
Step 9. Merge M and PerfP.

We are finished with Perf now, and M is the next step up on the HoP.
Because M is higher than Perf on the hierarchy of projections, features of M project.
Since M has no uninterpretable features to check, this is a maximal projection (MP).



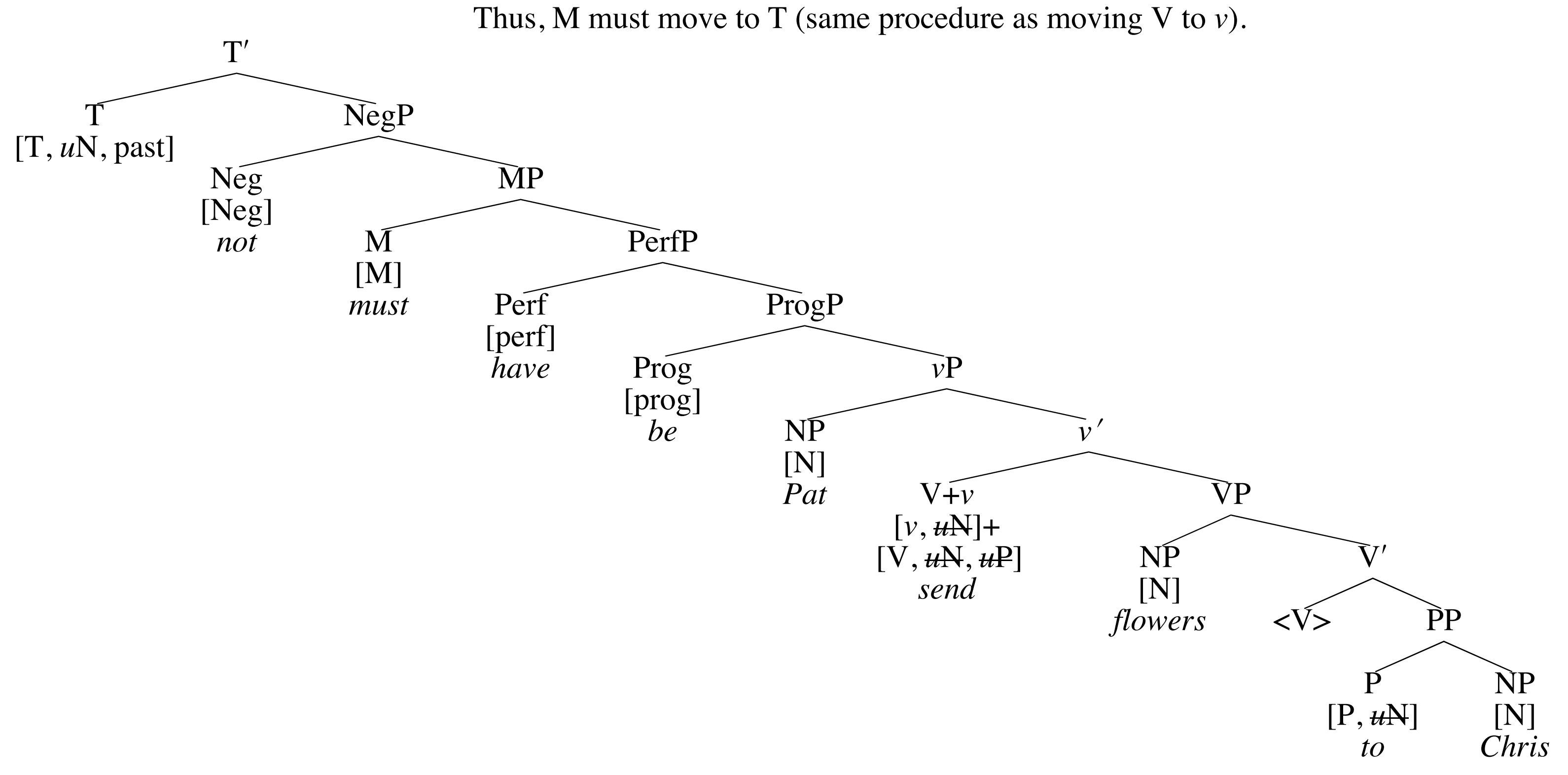
Step 10. Merge Neg and MP.

We are finished with M now, and Neg is the next step up on the HoP.
Because Neg is higher than M on the hierarchy of projections, features of Neg project.
Since Neg has no uninterpretable features to check, this is a maximal projection (NegP).



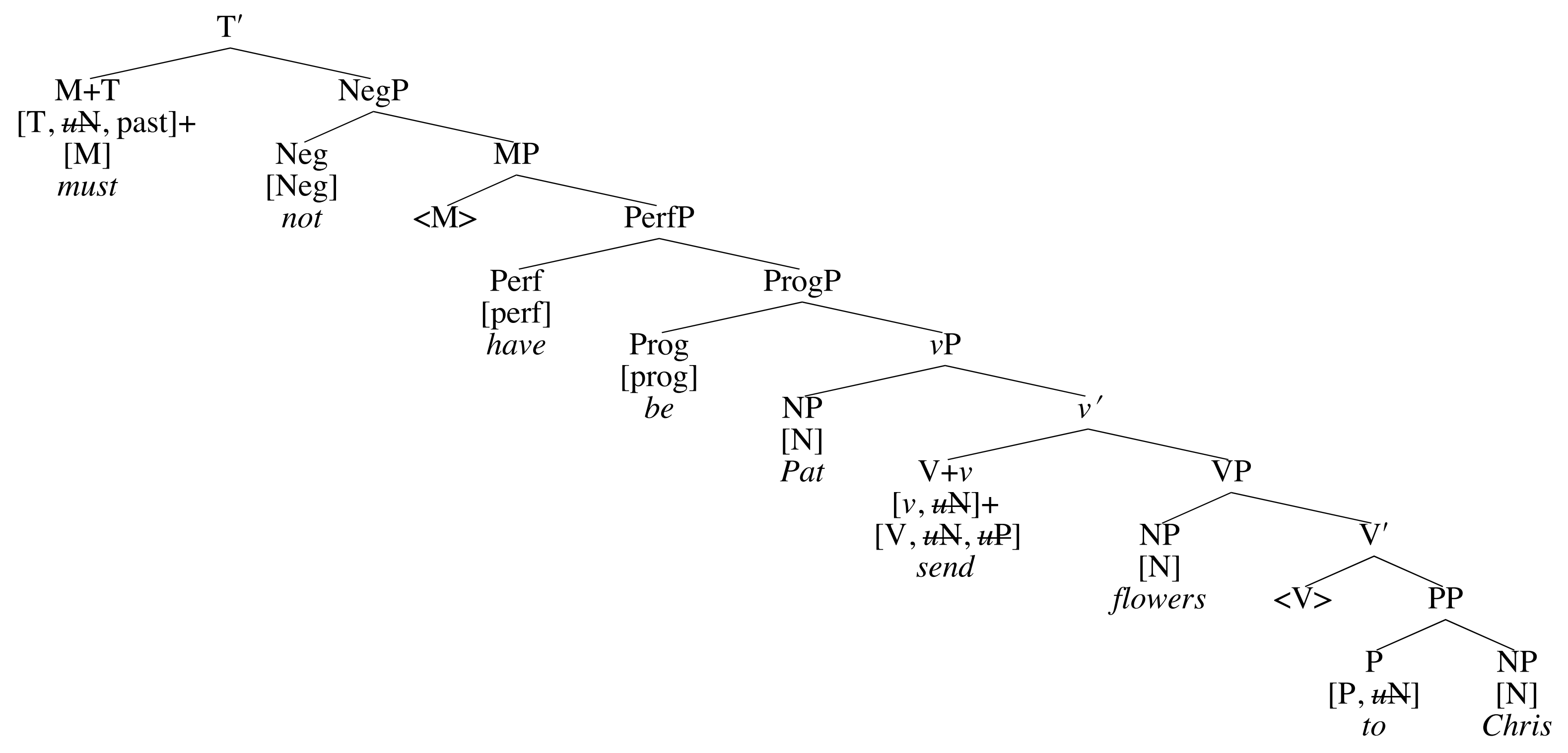
Step 11. Merge T and NegP.

We are finished with NegP now, and T is the next step up on the HoP.
Because T is higher than NegP on the hierarchy of projections, features of T project.
Since T has an uninterpretable feature to check, this is an intermediate projection (T′).
Thus, M must move to T (same procedure as moving V to *v*).



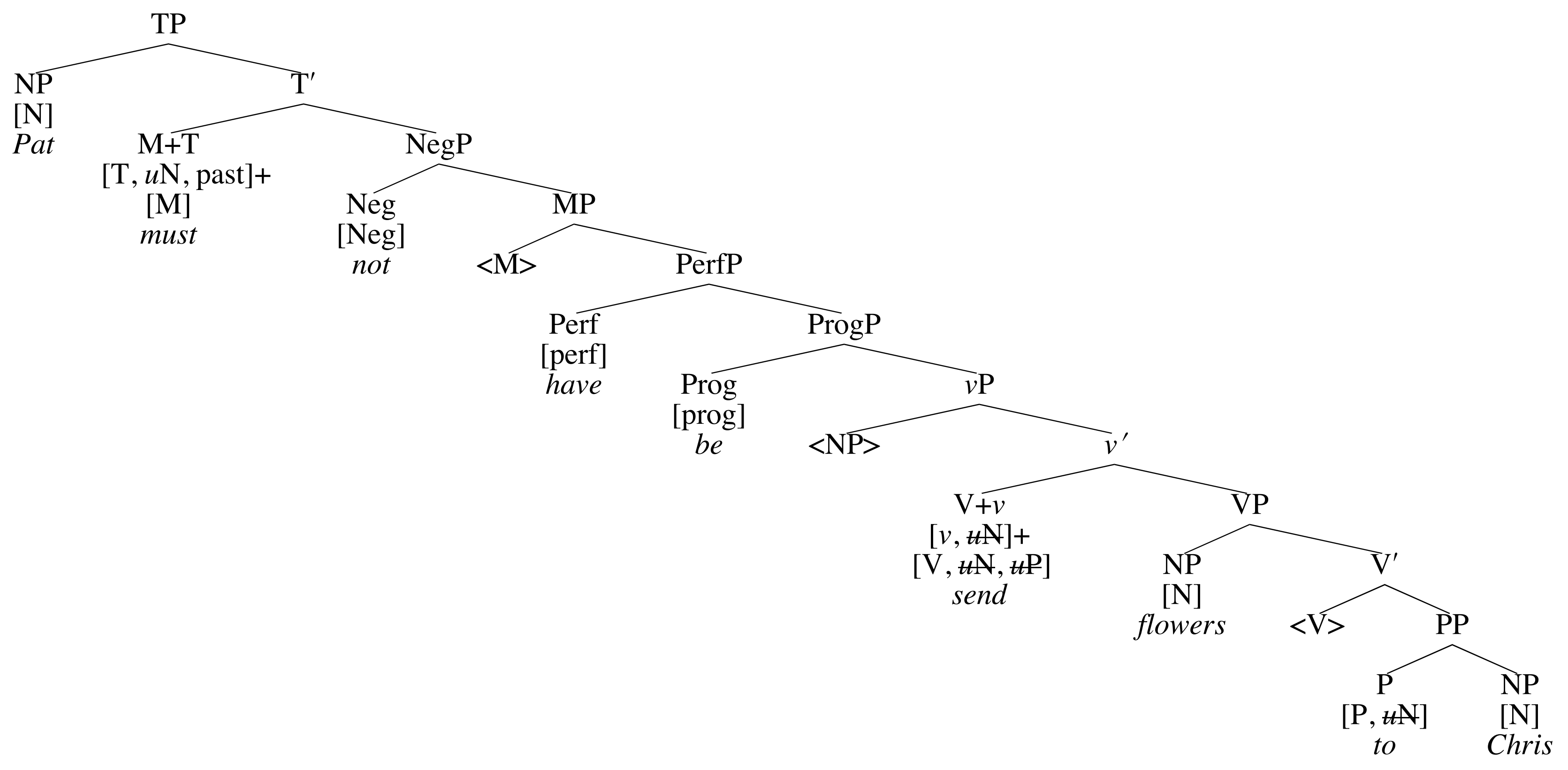
Step 12. Move M to T

M must move to T (same as moving V to *v*: written as M+T and with original <M> in brackets).
As with V and *v*, I will write the features of both M and T, with the features of T on top.
This is because it is the closest auxiliary (i.e., one of M, Perf, Prog) to T, and when there is an auxiliary, the topmost one always moves to T (with a couple of caveats to be discussed in a future class).



Step 13. Move *Pat* to the specifier of T.

T still has an uninterpretable [*u*N] feature to check.
Nothing remains in the numeration that can check this feature.
The [*u*N] feature of T matches the [N] feature of *Pat*. (T c-commands *Pat*, and there is nothing closer to T with an [N] feature).
Pat is moved (copied, and Merged with T′).
This puts the remaining [*u*N] feature of T close to the [N] feature of *Pat*. [*u*N] is checked.
Since T had its feature checked, the features of T project.
Since there are no more uninterpretable features on T, this is a maximal projection (TP).



The derivation is done. No uninterpretable features remain. Pronunciation: *Pat must not have been sending flowers to Chris*.
To determine the pronunciation (again, this will be made more formal in class shortly after the midterm):
Each of M, Perf, Prog, and *v* need an inflectional ending. Which inflectional ending it is is determined by what's just above it.
T (originally above M) is [past] and so the form of M is *must* (rather than whatever its present form would have been.)
M is above Perf, and M provides a bare form "ending." So, Perf is pronounced *have*.
Perf is above Prog, and Perf provides the –*en* ending. So, Prog is pronounced *been*.
Prog is above *v*, and Prog provides the –*ing* ending. So, *v* (or rather V+*v*) is pronounced *sending*.