CAS LX 522 Syntax I
Fall 2015

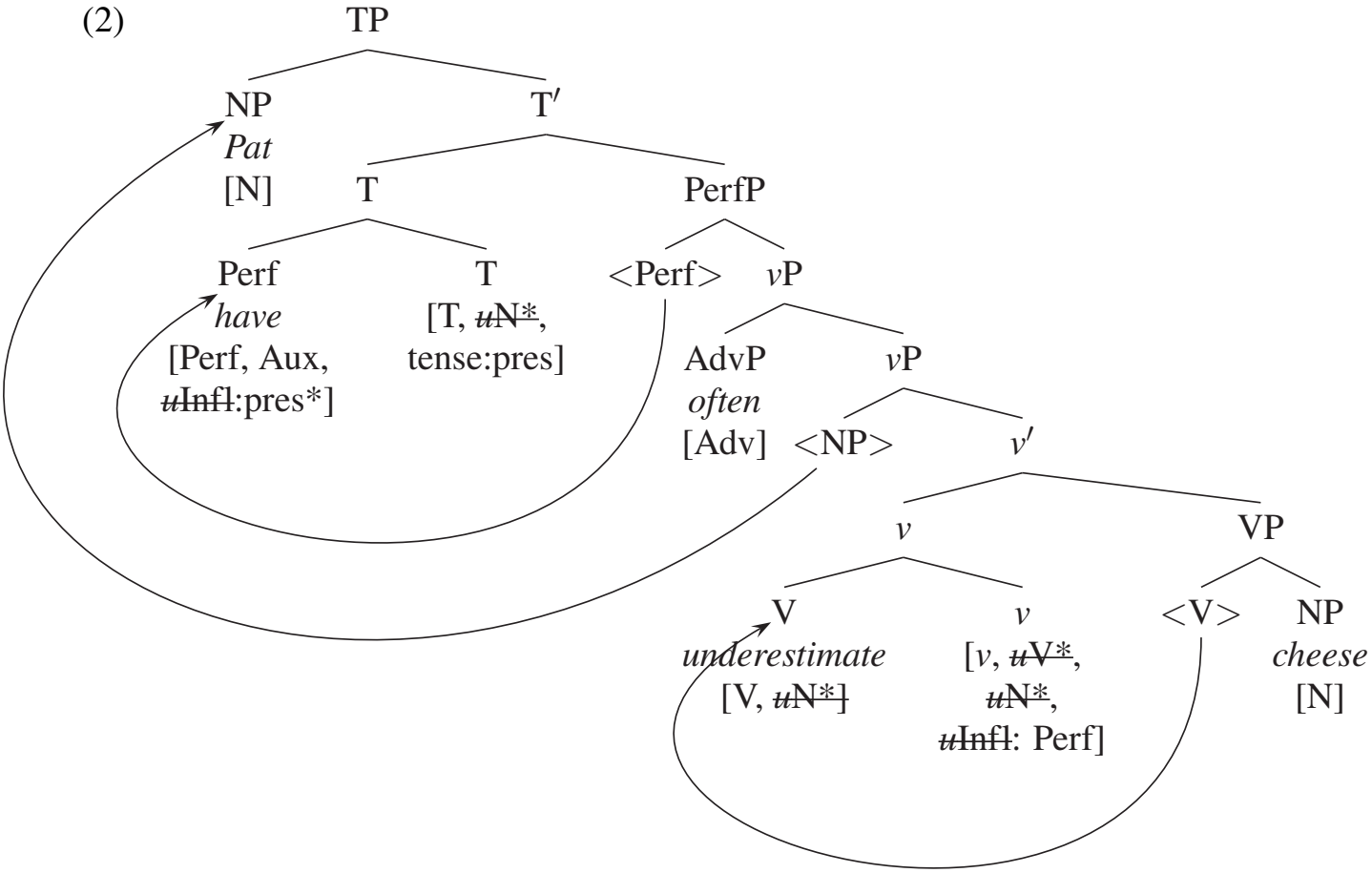
Homework #5
DUE THU OCT 8

Handling the infinitival *to*…
This homework has a unique status, in that it's a bit like a required reading. In section 3 are going to be working out a way to think of the infinitive marker to. *I will guide you to an analysis. However, regardless of whether you successfully reach the end, this will be the analysis that we use from here on out, so it is important that you understand how it works by reading the key afterwards.*

1 Practice using Agree

(1) Pat has often underestimated cheese.

(2)



And here are some questions about the the tree in (2). These are not supposed to be difficult. They are supposed to start us off together.

Locate the slide on page 4 of handout 10 that has Agree as its title. It has things called X, Y, F1, F2 in an *if—then* structure.

First step. In the first step, the V and the NP *cheese* are combined by Merge. This is allowed because an uninterpretable feature is checked. V has a [*u*N*] feature, NP has

an [N] feature. This Merge checks the (strong) uninterpretable [*u*N*] feature because it satisfies the conditions on Agree:

- V has feature [*u*N*], NP has feature [N].
- V c-commands NP *and* NP c-commands V (after the Merge).
- [*u*N*] is uninterpretable.
- [*u*N*] matches [N].
- [*u*N*] and [N] are close enough: There is no closer matching feature between V and NP, and, although [*u*N*] is strong, V and NP share the same mother node (after the Merge).

The result is that:

- Any unvalued feature would be valued (though there aren't any).
- The uninterpretable feature ([*u*N*]) is checked.

1. Agreement with Perf. Now, consider the derivation a few steps later, when Perf has been Merged with *v*P in order to satisfy the Hierarchy of Projections. Although by this point the strong features of *v* have been checked, *v* still has an uninterpretable feature, [*u*Infl:]. Run through the definition of Agree, just as I did above for the first step, except now using Perf as X and *v* as Y, in order to demonstrate that Agree will result in checking this feature. And by "just as I did above for the first step," I mean provide a bullet point for each condition in the *if* clause of the slide, and for each result in the *then* clause. It's tedious, but you have exactly two to do, this one being the first, the next task below being the second. Your survival is pretty much guaranteed.

2. Matching after Merging T. The next step is to Merge T. Perf still has a [*u*Infl:] feature to check. Run through the first four points of the definition of Agree, again following the model I gave in the first step, to demonstrate that at this point, [tense:pres] on T matches the [*u*Infl:] feature of Perf. Use T as X and Perf as Y.

3. Valuing after Merging T. Because [tense:pres] on T matches the [*u*Infl:] feature of Perf, the unvalued feature is valued. Write the newly valued [*u*Infl:] feature (by filling in the value). Take special note of the point made on handout 10 in the last "Auxiliaries moving to T" slide on page 6. This is a [*u*Infl:] that is valued by T.

4. Not checking the feature. Now that the feature has been valued, look at the fifth part in the definition of the conditions under which Agree happens ("If…X and Y are close enough…"). The [tense:pres] feature is unable to check the uninterpretable feature you just wrote. Why? (It is the fact that checking cannot be accomplished here that will force Perf to head-adjoin to T.)

5. Checking the feature. The next step is to head-adjoin Perf to T. Now that Perf is adjoined to T, re-evaluate the fifth step. Why do [tense:pres] and the uninterpretable feature now count as "close enough"? (This is really basically trivial given what you answered just above—this fixes the problem that you identified in the previous task.)

2 Drawing some trees on your own

Your turn. Now, draw trees for (3) and (4), using the model from (2). **Ground rules for drawing the trees:**

- We now know how to use Agree, so you need to show the uninterpretable agreement features ([*u*Infl:]) and how they're valued.
- You *do not* need to show each step. Show the tree in its *final* form. That means: show everything that moves in the location it has moved to.
- Draw arrows indicating the movements.
- Draw angled brackets ($< >$) around the traces.
- If you move a head, draw the complex head that results.
- We now differentiate between strong and weak uninterpretable features.
- C-selection features are strong.
- *Do all of this stuff*—points will be taken off for not following the directions.

(3) Pat might have misunderstood Chris.

(4) Chris was not reciting morbid poetry.

3 Infinitival complements

In this exercise, we are going to extend the system we have to allow for sentences such as (5). Of particular relevance is the fact that it contains two clauses. One is the one whose verb is *expected*, and the other is the one whose verb is *to make*. The definition of "clause" here that I'm using is basically something that has a verb and a T node.

(5) I expected Tracy to make pasta

Although we cannot completely analyze embedded clauses yet (because we still need to discuss C, so we aren't using C yet), we can get a start on them by thinking about embedded infinitival clauses. An *infinitival clause* is a clause that is untensed, it is neither present nor past. To handle these, we will start by adding the following things to our system:

- Verbs like *want* or *expect* that take infinitival complements have a [*u*T*] feature—their "object" is a TP.
- We will call the θ-role of an embedded clause "Proposition."
- So, we add to the UTAH: TP sister of V is a Proposition.
- Infinitival clauses have a T with a feature [tense:inf].

It is common to start off thinking of *to* as itself being of category T. But this can't be right. Consider sentence (6).

(6) I wanted Tracy not to have been making pasta.

Part 1. Explain (concisely) why *to* cannot be T in (6). Assume that *not* cannot move. Assume the Hierarchy of Projections we used in class:

T > (Neg) > (Modal) > (Perf) > (Prog) > *v* > V

Part 2. Assume that *to* has the same category as some existing type (it's not a new type of thing). Propose a category for *to*, and use the sentences in (7) and (8) as part of a (quick, short) argument for how *to* behaves in at least one respect like other things of this category.

(7) a. * Tracy should can make pasta.
b. * Tracy can should make pasta.

(8) a. * I expect Tracy to should make pasta.
b. * I expect Tracy should to make pasta.

Part 3. Even in light of what you proposed in part 2, there is something strange about *to*. Compare (9) and (6)—what is different about *to* syntactically? (When thinking about this, it may help to assume that the [tense:inf] T—by itself—has no pronunciation.)

(9) Tracy should not have been making pasta.

Even in languages where there is not a direct analog of English *to*, it is very uncommon for verbs or auxiliaries to move to T in infinitival clauses—even when such things move to T in tensed clauses. Think about what it is in our system that makes auxiliaries move to T in tensed clauses (you might refer back to task 3 of the first section). Assume that the [tense:inf] feature of T in infinitival clauses can value a [*u*Infl:] feature below it, like [tense:pres] and [tense:past] can.

Part 4. The part of our system that causes auxiliaries to move to T is that part on page 6 of handout 10 that you looked at earlier: If two conditions hold, then a feature is valued as strong. Modify the second condition in such a way that it can account for the behavior of *to* (and fits in with the crosslinguistic tendency just mentioned). There are a couple of possible changes you could make, pick one.

Part 5. Draw a tree for (10) (same ground rules as for the trees you drew in section 2). Assume that non-finite (infinitival) T is silent (has no pronunciation).

(10) I wanted Tracy not to make pasta.