

What have we got so far?

We do have quite a lot of categories, namely

- A - for Adjectives
- N - for Nouns
- P - for Prepositions
- V - for Verbs
- D - for Determiners
- C - for Complementizers
- T - for Tense (might as well be called I)

not too different from other approaches, but then here we go

- v - little v, and since I just don't get the whole cause-whatever-talk, the main-reason I see for its being there is the dire need for strict configurational positions the whole approach is in (I do not, however, want to say that there is no motivation for it – I just don't get this go-cause equals give or move stuff) note that there are assumed to be at least 2 different little vs (as far as I can remember, that is one for 'normal' verbs and one for unergatives)
- n - I think we all agreed that this really IS strange; again, the motivation for n's being there is that it provides positions we might use for theta-assignment

and then, we do have all those 'aspectual'-heads which come as one-element-categories, and the funny part about those one element sets is that their only element is – an empty element (that, however, carries quite a lot of information)

- Poss - For Possessors; the Possessor is merged as its specifier, and the possessed is not merged at all – hierarchy of projection is the keyword
- Perf - For Perfect – carries a perf feature
- Prog - For Progressive-Tense, carries a prog feature
- Neg - For Negation

As to the operations, we only seem to need six concepts (now that is slim, isn't it)

- Merge - always triggered by satisfying a c-selectional feature → Check
- Adjoin - kind of bizarre operation that does not seem to need any justification
- Move - always triggered by satisfaction of locality, required by so-called strong-features → Agree(?)
- Agree - used for feature valuation; basically same as checking, but not under sisterhood
- Check - Checking under sisterhood, uninterpretable feature deletion; triggers Merge

I went through the text and tried to pick out all those 'general' principles

- Full Interpretation:** The structure to which the semantic interface rules apply contains no uninterpretable features. (p. 85)
- Checking Requirement:** Uninterpretable (c-selectional) features must be checked, and once checked, they delete. (ibid.)
- Checking under Sisterhood:** An *uninterpretable c-selectional feature F* (uF) on a syntactic object Y is checked when Y is sister to another syntactic object Z which bears a *matching feature F* (F)

- Merge:**
1. Merge applies to two syntactic objects to form a new one.
 2. The new syntactic object is said to contain the original ones, which are sisters but are not linearized.
 3. Merge only applies to the root nodes of syntactic objects.
 4. Merge allows the checking of an uninterpretable c-selectional feature on a head, since it creates a sisterhood syntactic relation.

since we have Merge in place, let's define our Heads in a new fancy way:

Definition of Head: The head is the syntactic object which selects in any Merge operation.¹

as for Heads, there is this nice mnemonic:

The item that projects is the item that selects.

Extension Condition: A syntactic derivation can only be continued by applying operations to the root projection of the tree.

Θ-Generalization: Each Θ-role must be assigned but a constituent cannot be assigned more than one Θ-role. – now that is an old one

Now we need some definitions for our old friends complement (argument), adjunct and specifier – pretty straightforward

First Merge → Complements

Complements are typically assigned Θ-roles.

Definition of a maximal: Syntactic objects which have no c-selectional features to be checked are called maximal, since they project no further.

Definition of a phrase: A maximal syntactic object of category X is abbreviated XP, is said to be a phrase of category X.

Definition of minimal: A minimal projection is just a lexical item.²

head-complement structure: Head-complement structures arise from the first application of Merge to satisfy a selectional feature of a head.

Note that merge is totally indifferent when it comes to word order that is the structure resulting from merging X with Y is the same as that resulting from merging Y with X. What regulates word-order (linearization to me seems just to be another fancy term) is parametrized(?) not with respect to merge but with respect to complement and specifier, respectively. In English, complements are **linearized** right to the head and specifiers are **linearized** left to the head – in Japanese, for example, complements are linearized left to the head.

Second Merge → Specifier

¹ I have been wondering whether 'the syntactic object' also allows for not maximally-projected phrases to speak of them as Heads (since they usually bear those features still in need for satisfaction) – this seems, however, to be a purely terminological matter

² Not clearly stated but I don't think I got this wrong – cf. p.106

Specifiers may also be assigned Θ -roles.

This results in quite a simple phrase-structure:

We have „three levels of projection [for the verb – the same is later argued to also be true for the clausal and the nominal domain]: we have the lexical item itself (the minimal projection, written as X^{\min} [...]). The second level is the maximal projection, which results when all selectional features have been checked (written as XP [...]). Finally, we have an intermediate projection, sometimes called the bar-level projection, and written as X' [...].“³

So we might make the following explicit statement:

Complements are (usually) merged with the minimal projection, resulting in an intermediate projection.

Specifiers are merged with the intermediate projection, resulting in a maximal projection (that is, a fully fledged phrase)

Adjoin → **Adjunct**

This is really unsatisfactory – I do not think, however, that this is Adger's fault; Adjunction seems to be a phenomenon very hard to capture, and while I do think that the ideas presented by Neven (cartographic they were called, I think?) are quite interesting, I do not think they are very plausible. Back to the little Adger has to say, then:

„Adjuncts are elements that are somehow incorporated into a sentence, *but not via the checking of selectional features.*“⁴

„[A]djuncts are sisters of phrasal nodes“⁵

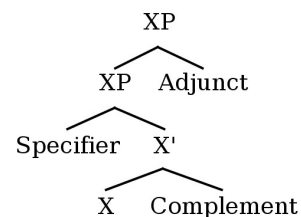
Adjuncts do not receive any Θ -roles.

Adjoin: „[...] another basic operation [...] which, unlike Merge, does not need to be triggered. [...] Adjoin inserts a phrasal object (the Adjunct) into another phrasal object at its outermost level. *It does not create a new object*, it expands one of the old ones by stretching its outermost layer into two parts and inserting the adjoined object between them.“ (is it just me or does this sound like the splitting thing Neven is always talking about?)

Complements are sisters of lexical items (X)

Specifiers are sisters of X' nodes

Adjuncts are sisters of XP nodes



Agree An uninterpretable feature F on a syntactic object Y is checked when Y is in a c-command relation with another syntactic object Z which bears a matching feature F.

³Sloppily cited from p. 109

⁴p.111

⁵ibid.

This leaves open the question whether it is the interpretable feature bearer (that is Z) which needs to c-command Y or the other way round or whether both possibilities are allowed. One page later, however, Adger seems to me to make it explicit:

Agree In a configuration (X[F: val] c-command Y[uF:]), F (on X) might check and value (actually I think it should be the other way round) uF on Y, resulting in (X[F:val] c-command Y[~~u~~F: val])

That is, it is Z (in the old definition, X in the one above) that needs to c-command Y, and not the other way round. This might answer our questions regarding 'upward-feature-valuation' – it is ruled out since Agree does not allow for it.

Adger calls this checking by valuing, so perhaps it's best not to say that F checks and values uF but rather that F values uF, thereby checking it. Since, however, the notion of strong features and the principle of locality (that is, checking under sisterhood) is needed to ensure certain word-order-phenomena one might want to keep valuing and checking separate all the time (at least that's what I want to do).

Strong features Certain features are strong features, that is they give rise to movement phenomena. A strong feature always needs to be checked under sisterhood; strong features are not, however, satisfied by Merging but by Adjoining with a phrase already somewhere in the tree (I think there needs to be c-command relation, that is, the features need to Agree; this is, however, problematic if one wants to hold that it is always the interpretable feature that needs to c-command the uninterpretable – since Adger never claims this to be the case explicitly, no problem for him, but the question about upward-valuing then is back) that is moved.