CAS LX 522
Syntax I

Episode 6a. Parametric differences and do-support
5.5-5.6

Recap: features
- The **lexicon** contains bundles of features. These feature bundles are assembled by a computational process into syntactic structures for interpretation by the conceptual-intensional and articulatory-perceptual systems.
- Among these features, we have
  - **Interpretable features** (such as the category feature that determines the category of the lexical item)
  - **Uninterpretable features** (such as the selectional feature [uN] on a transitive verb). Uninterpretable features are intolerable at the interfaces, and must be removed (by checking) or the derivation crashes.

Recap: uninterpretable features
- **Uninterpretable features** vary along two dimensions. **Privative/unvalued : strong/weak**.
  - **Privative features** (such as [uN]) which are checked by matching features (such as [N] or [uN]).
  - **Unvalued features** (such as [unfl:]) which are checked by features that can provide a value (such as [tense:past]).
  - **Strong** uninterpretable features can only be checked if they are local (sister) to the feature that checks them.
  - **Weak** uninterpretable features can be "checked at a distance."
- **Strong features** can force movement, but because the system is economical (lazy), no movement is allowed just to check a weak feature.

Recap: Matching and Checking
- **Checking** relates an uninterpretable feature and a **matching** feature, allowing the uninterpretable feature to be ignored at the interface.
- If the uninterpretable feature is strong, the matching feature must be **local** (e.g., a feature of the sister) in order for the uninterpretable feature to be checked.
  - For [uV*] on v, it matches the [V] feature of the verb below it, then the verb must move up to v to check [uV*].
  - For [unfl:] on an auxiliary, the [tense:past] feature (above it) matches it and values it as strong (in English), then the auxiliary must move up to T for the feature to be checked.

Recap: Agree
- If:
  - X has feature [F1], Y has feature [F2]
  - X c-commands Y or Y c-commands X
  - [F1] and/or [F2] are/is uninterpretable.
  - [F1] matches [F2]
  - X and Y are close enough, meaning:
    - There is no closer matching feature between X and Y.
    - If [F1] or [F2] is strong, X and Y share the same mother node
- Then:
  - Any unvalued feature ([F1] or [F2]) is valued.
  - The uninterpretable feature(s) is/are checked.

Recap: Merge
- **Merge**: create a new syntactic object from two existing syntactic objects, with the label (features) projecting from one. Merge happens:
  - To **check an uninterpretable feature**: the label of the one with the uninterpretable feature projects.
    - Example: c-selection features, such as the [uN*] feature of P.
  - To satisfy the Hierarchy of Projections: the label of the higher one in the hierarchy projects and no features are checked.
    - This only happens once all of the strong uninterpretable features in the non-projecting object have been checked (and any adjunctions to be done have been done)
Recap: Adjoin, Agree, HoP

- **Adjoin** is like Merge, but it does not result in the checking of a feature.
  - Merge always takes priority over Adjoin, so Adjoin only happens once the (strong) uninterpretable features of the object being adjoined to are checked.
  - Adjoining YP to XP results in another XP (the maximal projection is extended), so YP becomes in essence both a daughter and a sister to XP.
- **Agree** is the operation that checks (and values where appropriate) features under c-command.
- **Hierarchy of Projections:**
  \[ T > (\text{Neg}) > (\text{M}) > (\text{Perf}) > (\text{Prog}) > v > V \]

Recap: Move

- There are two basic kinds of movement.
  - One is **head-movement**, where a head moves up to join with another head.
    - Examples: V moves to v, Perf moves to T
  - The other is **XP-movement**, where a maximal projection moves up to a specifier of a higher phrase.
    - Example: The subject moving to SpecTP.
- Both happen because a strong uninterpretable feature needs to be checked.

Recap: UTAH

- The Uniformity of Theta-assignment Hypothesis determines the \( \theta \)-role of an argument based on its position in the structure.
  - NP daughter of vP: Agent (\( v_{\text{Agent}} \))
  - NP daughter of vP: Experiencer (\( v_{\text{Experiencer}} \))
  - NP daughter of VP: Theme
  - PP daughter of V: Goal
  - NP daughter of V: Possessee
  - TP sister of V: Proposition

French vs. English

- In English, adverbs cannot come between the verb and the object.
  - *Pat eats often apples.
  - Pat often eats apples.
  - In French it’s the other way around.
  - Jean mange souvent des pommes.
    - Jean eats often of the apples
  - *Jean souvent mange des pommes.
  - If we suppose that the basic structures are the same, why might that be?

V raises to T in French

- What it looks like is that both V and auxiliaries raise to T in French.
- This is a **parametric difference** between English and French.
- A kid’s task is to determine whether V moves to T and whether auxiliaries move to T.

<table>
<thead>
<tr>
<th>T values [\text{incl}] on Aux</th>
<th>T values [\text{incl}] on ( v )</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>Strong</td>
</tr>
<tr>
<td>French</td>
<td>Strong</td>
</tr>
</tbody>
</table>
Jean (n’) appelle pas Marie

First, build the vP just as in English.
- Merge appelle and Marie to form the VP. Merge v and VP to satisfy the HoP, move V to adjoin to v to check v’s [uV*] feature. Merge Jean and v’.

\[ T \quad \begin{array}{c}
\text{Neg} \\
\text{pas}
\end{array} \quad \begin{array}{c}
\text{Jean} \\
\text{[N]}
\end{array} \quad \begin{array}{c}
\text{appelle} \\
\text{[V]}
\end{array} \quad \begin{array}{c}
\text{Marie} \\
\text{[N]}
\end{array} \]

Merge Neg with vP to form NegP (following the HoP).

\[ T \quad \begin{array}{c}
\text{Neg} \\
\text{pas}
\end{array} \quad \begin{array}{c}
\text{Jean} \\
\text{[N]}
\end{array} \quad \begin{array}{c}
\text{appelle} \\
\text{[V]}
\end{array} \quad \begin{array}{c}
\text{Marie} \\
\text{[N]}
\end{array} \]

Jean (n’) appelle pas Marie

- Merge T with NegP to form T’ (again, following the HoP).
- Now T with its [tense:pres, uN*, …] feature c-commands v and its [uInfl] feature. They Match. But in French, when [uInfl] on v is valued by T it is strong. So…

\[ T \quad \begin{array}{c}
\text{Neg} \\
\text{pas}
\end{array} \quad \begin{array}{c}
\text{Jean} \\
\text{[N]}
\end{array} \quad \begin{array}{c}
\text{appelle} \\
\text{[V]}
\end{array} \quad \begin{array}{c}
\text{Marie} \\
\text{[N]}
\end{array} \]

v has to move to T. Notice that at this point v has V adjoined to it. You can’t take them apart. The whole complex head moves to T.

\[ T \quad \begin{array}{c}
\text{Neg} \\
\text{pas}
\end{array} \quad \begin{array}{c}
\text{Jean} \\
\text{[N]}
\end{array} \quad \begin{array}{c}
\text{appelle} \\
\text{[V]}
\end{array} \quad \begin{array}{c}
\text{Marie} \\
\text{[N]}
\end{array} \]

Jean (n’) appelle pas Marie

- And then, we move the subject up to SpecTP to check the final uninterpretable (strong) feature of T, [uN*].

\[ T \quad \begin{array}{c}
\text{Neg} \\
\text{pas}
\end{array} \quad \begin{array}{c}
\text{Jean} \\
\text{[N]}
\end{array} \quad \begin{array}{c}
\text{appelle} \\
\text{[V]}
\end{array} \quad \begin{array}{c}
\text{Marie} \\
\text{[N]}
\end{array} \]

So, French is just like English, except that even v moves to T.

\[ T \quad \begin{array}{c}
\text{Neg} \\
\text{pas}
\end{array} \quad \begin{array}{c}
\text{Jean} \\
\text{[N]}
\end{array} \quad \begin{array}{c}
\text{appelle} \\
\text{[V]}
\end{array} \quad \begin{array}{c}
\text{Marie} \\
\text{[N]}
\end{array} \]

Swedish

- Looking at Swedish, we can see that not only do languages vary on whether they raise main verbs to T, languages also vary on whether they raise auxiliaries to T:
  - ...om hon inte har köpt boken
  - "...whether she has bought the book."
  - ...om hon inte köpte boken
  - "...whether she didn’t buy the book."

- So both parameters can vary.
  - Remember the light box: By saying these were parameters, we predicted that we would find these languages.
Typology of verb/aux raising

- Interestingly, there don’t seem to be languages that raise main verbs but not auxiliaries.
  - This double-binary distinction predicts there would be.
  - It overgenerates a bit.
- This is a pattern that we would like to explain someday, another mystery about Aux to file away.
- Sorry, we won’t have any satisfying explanation for this gap this semester.

<table>
<thead>
<tr>
<th></th>
<th>T values ([\text{unfl:}]) on Infl</th>
<th>T values ([\text{unfl:}]) on v</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>Strong</td>
<td>Weak</td>
</tr>
<tr>
<td>French</td>
<td>Strong</td>
<td>Strong</td>
</tr>
<tr>
<td>Swedish</td>
<td>Weak</td>
<td>Weak</td>
</tr>
<tr>
<td>Unattested</td>
<td>Weak</td>
<td>Strong</td>
</tr>
</tbody>
</table>

Irish

- In Irish, the basic word order is VSO (other languages have this property too, e.g., Arabic).
- Phéag Mháire an lucharachán.
  - kissed Mary the leprechaun
  - Mary kissed the leprechaun.
- We distinguish SVO from SOV by supposing that the head-complement order can vary from language to language (heads precede complements in English, heads follow complements in Japanese).
- We may also be able to distinguish other languages (OVS, VOS) by a parameter of specifier order.
- But no combination of these two parameters can give us VSO.

Irish

- But look at auxiliary verbs in Irish:
  - Tá Mháire ag-pógáil an lucharachán.
  - Is Mary ing-kiss the leprechaun
  - Mary is kissing the leprechaun.
- We find that if an auxiliary occupies the verb slot at the beginning of the sentence, the main verb appears between the subject and verb:
  - **Aux S V O**.
- What does this suggest about
  - The head-parameter setting in Irish?
  - How VSO order arises?

French vs. Irish

- Remember this step in the French derivation before?
  - I’ve omitted negation to make it simpler.
- What if we stopped here?
  - In French it would crash (why?).
  - But what if it didn’t crash in Irish?
  - What would have to be different?

SVO to VSO

- Irish appears to be essentially an SVO language, like French.
- Verbs and auxiliaries raise past the subject to yield VSO.
- We can analyze the Irish pattern as being minimally different from our existing analysis of French—just one difference, which we hypothesize is another parametric difference between languages.
- V and Aux both raise to T (when tense values the [\text{unfl:}] feature of either one, [\text{unfl:}] is strong) in Irish, just as in French.

Irish

<table>
<thead>
<tr>
<th></th>
<th>T values ([\text{unfl:}]) on Infl</th>
<th>T values ([\text{unfl:}]) on v</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>Strong</td>
<td>Weak</td>
</tr>
<tr>
<td>French</td>
<td>Strong</td>
<td>Strong</td>
</tr>
<tr>
<td>Swedish</td>
<td>Weak</td>
<td>Weak</td>
</tr>
<tr>
<td>Unattested</td>
<td>Weak</td>
<td>Strong</td>
</tr>
</tbody>
</table>

Parametric differences

- We could analyze Irish as being just like French except without the strong [\text{unfl:}] feature on T.
  - Without that feature, the subject doesn’t need to move to SpecTP. The order would be VSO, or AuxSVO.
- So, languages can vary in, at least:
  - Head-complement order
  - (Head-specifier order)
  - Whether [\text{unfl:}] on Aux is strong or weak when valued by T
  - Whether [\text{unfl:}] on v is strong or weak when valued by T
  - Whether T has a [\text{unfl:}] feature or not
do-support

- In French, verbs move to T.
- In English, they don’t move to T.
- That’s because in French, when [tense:past] values [\text{\text{\footnotesize uInf}}] on v, it is strong, and in English, it is weak.
- What this doesn’t explain is why do appears sometimes in English, seemingly doing nothing but carrying the tense (and subject agreement).
- The environments are complicated:
  - Tom did not commit the crime.
  - Tom did not commit the crime, but someone did.
  - Zoe and Danny vowed to prove Tom innocent, and prove Tom innocent they did.
  - Tom (has) never committed that crime.

Technical difficulties

- How do we generally know to pronounce \( V^+v \) as a past tense verb?
  - T values the \([\text{\text{\footnotesize uInf}}]\) feature of v. The presumption is that \( v^+v \text{\footnotesize uInf:past} \) sounds like “ate.” And T doesn’t sound like anything.
  - But this happens whether or not \( v \) is right next to T. \( v \) still has a \([\text{\text{\footnotesize uInf}}]\) feature that has to be checked.
  - So, the questions are, how do we:
    - Keep from pronouncing the verb based on v’s \([\text{\text{\footnotesize uInf}}]\) feature if T isn’t right next to it?
    - Keep from pronouncing do at T if \( v \) is right next to it?
  - We need to connect T and v somehow.

Technical difficulties

- Let’s be creative: Suppose that the tense features on v (the value of the \([\text{\text{\footnotesize uInf}}]\) feature) “refer back” to the tense features on T.
  - Agree can see relatively far (so T can value the \([\text{\text{\footnotesize uInf}}]\) feature of v, even if it has to look past negation).
  - But “referring back” is more limited, basically only available to features that are sisters. Negation will get in the way for this.
  - So if you try to pronounce tense on v but T is too far away, the back-reference fails, and v is pronounced as a bare verb. But the tense features have to be pronounced somewhere, so they’re pronounced on T (as do).

Technical difficulties

- The connection between T and v is that (when there are no auxiliaries), T values the \([\text{\text{\footnotesize uInf}}]\) feature of v.
  - This sets up a relationship between the two heads.
  - Adger calls this relationship a chain.
  - We want to ensure that tense features are pronounced in exactly one place in this chain.
    - If the ends of the chain are not close enough together, tense is pronounced on T (as do). If they are close enough together, tense is pronounced on \( v^+v \).

Technical difficulties

- Adger’s proposal:
  - Pronouncing Tense Rule (PTR)
    - In a chain (T[tense], v[\text{\text{\footnotesize uInf}}:\text{\text{\footnotesize tense}}]), pronounce the tense features on v only if v is the head of T’s sister
    - NegP, if there, will be the sister of T (HoP), but Neg has no \([\text{\text{\footnotesize uInf}}]\) feature. do will be inserted.
    - Adverbs adjoin to vP, resulting in a vP, v has an \([\text{\text{\footnotesize uInf}}]\) valued by T and adverbs don’t get in the way of vP being the sister of T. Tense is pronounced on the verb (v).
    - If vP is gone altogether, do is inserted.
Pat did not call Chris

- So, here, T and v form a chain because [tense:past] valued [unif:past]. But v is not the head of T's sister.

**The Big Picture**

- Do-support comes to the rescue. What this means is just that T is pronounced as do with the tense specifications on T. According to PTR, we don't pronounce them on v. The tree doesn't change.

**History**

- In trying to model what we know (since it isn't conscious knowledge) some of the first attempts looked like this (Chomsky 1957):
  - Phrase Structure Rules
    - S → NP (Aux) VP
    - VP → V (NP) (PP)
    - NP → (Det) (Adj+) N
    - Aux → (Tns) (Modal) (Perf) (Prog)
    - N → Pat, lunch, ...
    - Tns → Past, Present
    - Perf → have -en
    - Modal → can, should, ...
    - PP → at, in, to, ...
    - Prog → be -ing

- An S can be rewritten as an NP, optionally an Aux, and a VP. An NP can be rewritten as, optionally a determiner, optionally one or more adjectives, and a noun. ...

- What we know is that an S has an NP, a VP, and sometimes an Aux between them, and that NPs can have a determiner, some number of adjectives, and a noun.

---

Pat never called Chris

- If there is an adverb like never, PTR still allows tense to be pronounced on v (so T doesn't have any pronunciation of its own at all).

---

History

- Phrase Structure Rules
  - S → NP (Aux) VP
  - VP → V (NP) (PP)
  - NP → (Det) (Adj+) N
  - PP → P NP
  - Aux → (Tns) (Modal) (Perf) (Prog)
  - N → Pat, lunch, ...
  - Tns → Past, Present
  - Perf → have -en
  - Modal → can, should, ...
  - PP → at, in, to, ...
  - Prog → be -ing

- In this way, many sentences can be derived, starting from S.

- The tree-style structure is a way to record the history of the derivation from S to the words in the sentence.

- We model our knowledge of English as a machine that (ideally, when it's finished) will generate all of the sentences of English and no others.
History continues

- Through the 60s there were good people working hard, figuring out what kinds of phrase structure rules and transformations are needed for a comprehensive description on English.
- As things developed, two things became clear:
  - A lot of the PSRs look pretty similar.
  - There’s no way a kid acquiring language can be learning these rules.

So, Chomsky proposed: Aux → {Tns, Modal} (Perf) (Prog) Tns → Past, Present Modal → can, should, ...
Perf → have -en
Prog → be -ing
Past → -ed

Yielding something like this:

![Affix Hopping Diagram]

- If you build a sentence this way, things aren’t in the right order, but there’s a simple transformation that can be done to get it right.

- Empirically, tense, perfect have, and progressive be each control the form of the verbal element to their right.

Chomsky (1970) proposed that there actually is only a limited set of phrase structure rule types.
For any categories X, Y, Z, W, there are only rules like:
XP → YP X’
X’ → X’ WP
X’ → X ZP

First, a “deep structure” (DS) tree is built, however you like but:
- Selectional restrictions must be satisfied
- α-rules must be assigned
- Etc.

Then, adjustments are made to get the “surface structure” (SS)
- Things more or less like Affix Hopping, or moving V to v, or moving the subject to SpecTP.
- Further constraints are verified here:
  - is there a subject in SpecTP? Etc.

Finally, the result is assigned a pronunciation (PF), and, possibly after some further adjustments, an interpretation (LF).

Which brings us to 1993

- The most recent change in viewpoint was to the system we’re working with now (arising from the Minimalist Program for Linguistic Theory).
- The constraints that applied to the structures in GB were getting to be rather esoteric and numerous, to the extent that it seemed we were missing generalizations.
- The goal of MPLT was to “start over” in a sense, to try to make the constraints follow from some more natural assumptions that we would need to make anyway.

This new view has the computational system working at a very basic level, forcing structures to obey the constraints of GB by enforcing them locally as we assemble the structure from the bottom up.

Why is this better? It’s a further reduction to even more general principles. The idea is that you need a few things to construct a language-like system—and there’s nothing else.

X-bar theory

- If drawn out as a tree, you may recognize the kind of structures this proposal entails. These are structures based on the “X-bar schema”.
  - XP → YP X’
  - X’ → X’ WP
  - X → X ZP
  - WP being the “specifier”, ZP being the “complement”, Adjuncts were considered to have a slightly different configuration then.

Why is this better? The types of rules are much more constrained, AND it also makes predictions about structure and constituency that turn out to be more accurate.

X

WP

ZP

Which brings us to 1993

- Around 1981, the view shifted from thinking of the system as constructing all and only structures with PSRs and transformations to a view in which structures and transformations could apply freely, but the grammatical structures were those that satisfied constraints on [various stages of] the representation.

Why is this better? Most of the construction-specific rules were made to follow from more general principles, interacting, AND again, it caused us to look for predictions, which were better met.

GB

- An ancestor to the kinds of movement rules and of course the Agree operation we’ve been talking about.
Features and technology

- The use of features to drive the system (uninterpretable features force Merge, because if they are not checked, the resulting structure will be itself uninterpretable) is a way to encode the notion that lexical items need other lexical items.
- A comment about the technology here:
  - The operations of Merge, Adjoin, Agree, and feature checking, the idea that features can be interpretable or not (or strong or weak) are all formalizations of an underlying system, used so that we can describe the system precisely enough to understand its predictions about our language knowledge.

Features and the moon

- We can think of this initially as the same kind of model as this:
- The Earth and the Moon don’t compute this. But if we write it this way, we can predict where the Moon will be.
- Saying lexical items have uninterpretable features that need to be checked, and hypothesizing mechanisms (matching, valuing) by which they might be checked is similarly a way to formalize the behavior of the computational system underlying language in a way that allows us deeper understanding of the system and what it predicts about language.

The “Minimalist Program”

- The analogy with the gravitational force equation isn’t quite accurate, given the underlying philosophy of the MP.
- The Minimalist Program in fact is trying to do this:
- Suppose that we have a cognitive system for language, which has to interact with at least two other cognitive systems, the conceptual-intensional and the articulatory-perceptual.
- Whatever it produces needs to be interpretable in the vernacular of each of these cognitive systems for the representation to be of any use.
- Suppose that the properties of these external systems are your boundary conditions, your specifications.
- The hypothesis of the MPLT is that the computational system underlying language is an optimal solution to those design specifications. So everything is thought of in terms of the creation of interpretable representations.