**French vs. English**

- In English, adverbs cannot come between the verb and the object.
  1) *Pat eats often apples.
  2) Pat often eats apples.

- In French it's the other way around.
  3) Jean mange souvent des pommes.
  4) *Jean souvent mange des pommes.

- If we suppose that the basic structures are the same, why might that be?

**French vs. English**

- Similarly, while only auxiliaries in English show up before negation (not)...
  - John does **not** love Mary.
  - John has **not** eaten apples.

- ...all verbs seem to show up before negation (pas) in French:
  - Jean (n') **aime pas** Marie.
    Jean (ne) loves not Marie
    'Jean doesn't love Marie.'
  - Jean (n') **a pas** mangé des pommes.
    Jean (ne)has not eaten of.the apples
    'Jean didn't eat apples.'

**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.

<table>
<thead>
<tr>
<th>England</th>
<th>French</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td>Strong</td>
</tr>
<tr>
<td>Weak</td>
<td>Strong</td>
</tr>
</tbody>
</table>

**Zinédine (n’) aime pas Marco**

- First, build the vP just as in English.
- Merge aime and Marco to form the VP. Merge v and VP to satisfy the HoP.
- Move V to Adjoin to v to check v's [\u{V}^\u{\text{V}}] feature. Merge Zinédine and v'.

**Zinédine (n’) aime pas Marco**

- Merge Neg with vP to form NegP (following the HoP).
Zinédine (n’) aime pas Marco
- Merge T with NegP to form T’ (again, following the HoP).
- Now T with its [tense:pres] 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…

\[ T'[\text{tense:pres}, T, uN*, \ldots] \]

\[ \text{NegP} \]
\[ \text{Neg pas} \]
\[ vP \]
\[ \text{NP} \]
\[ Zinédine \]
\[ v \]
\[ \text{VP} \]
\[ \langle \text{V} > \] NP Marco

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:
  - \[ \ldots \text{om hon inte köpte boken} \]
  - \[ \ldots \text{om hon inte har köpt boken} \]
  - \[ \ldots \text{whether she didn’t buy the book’} \]
  - \[ \ldots \text{whether she hasn’t bought the book.’} \]
  - 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 smidge.
  - 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>Language</th>
<th>Aux [uInfl:]</th>
<th>v [uInfl:]</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

- But look at auxiliary verbs in Irish:
  1) Tá Máire 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?

**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 [uInfl:] feature of either one, [uInfl:] is strong) in Irish, just as in French.

**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?

  ![Diagram of French derivation]

**Parametric differences**

- We can analyze Irish as being just like French except without the strong [uN*] 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 [uInfl:] on Aux is strong or weak when valued by T
  
  - Whether [uInfl:] on v is strong or weak when valued by T
  
  - Whether T has a [uN*] feature or not.

  (Later, when we look at German, we’ll suggest a different analysis of Irish, but this will work for now.)

**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 [uInfl:] 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:
    
    1) Tom **did not** commit the crime.
    2) Tom did not commit the crime, but someone **did**.
    3) Zoe and Danny vowed to prove Tom innocent, and prove Tom innocent they **did**.
    4) Tom (has) never **committed** that crime.

- The environments are complicated:
  
  1) Tom **did not** commit the crime.
  2) Tom did not commit the crime, but someone **did**.
  3) Zoe and Danny vowed to prove Tom innocent, and prove Tom innocent they **did**.
  4) 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 [uInfl:] feature of v. The presumption is that eat +V[uInfl: 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 [uInfl:] feature that has to be checked.
  • So, the questions are, how do we:
    • Keep from pronouncing the verb based on v’s [uInfl:] 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

• The connection between T and v is that (when there are no auxiliaries), T values the [uInfl:] 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

• Let’s be creative: Suppose that the tense features on v (the value of the [uInfl:] feature) “refer back” to the tense features on T.
  • Agree can see relatively far (so T can value the [uInfl:] 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).

PTR

• Adger’s proposal:
  • Pronouncing Tense Rule (PTR)
    In a chain (T[tense], v[uInfl: 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 [uInfl:] feature. do will be inserted.
  • Adverbs adjoin to vP, resulting in a vP. v has a [uInfl:] 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 [uInfl: past]. But v is not the head of T’s sister.

Pat did not call Chris

• 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.
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).

Historical interlude

- Back in the days of yore, people hypothesized that Pat will charm snakes had a structure like this.
- The subject NP Pat was in the specifier of “IP” (what we call “TP”), and the VP contained only the verb charm and the object NP snakes.
- Pat got an Agent θ-role by being in SpecIP, even though the fact that there is an Agent θ-role to be had is determined by the verb down in the VP.

The students will all...

- This predicts the normal word order pretty well, and so it was hypothesized that the verb simply assigned one of its θ-roles directly to SpecIP.
- No big deal, syntax works in strange and mysterious ways.
- At a certain point, someone started thinking about sentences like these:
  1) All the students will take the exam.
  2) The students will all take the exam.
- It’s fairly clear here that all the students is an NP, that it forms a coherent unit, a coherent concept. All really belongs with the students.

Floating quantifiers

1) All the students will take the exam.
2) The students will all take the exam.
- Back in the even older days, the hypothesis was that there was a special rule that turned the first sentence into the second.
- The Quantifier Float rule would move all over to the right, next to the VP.
- all NP … VP → NP … all + VP

Only some quantifiers float

- Quantifiers: every, some, all, most, several, many, both, four, …
  1) Every student will take the exam.
  2) *Student will every take the exam.
  3) Several students will take the exam.
  4) *Students will several take the exam.
- It works for both and all:
  5) The students will both take the exam.
  6) The students will all take the exam.
- What’s a difference between every, some, several, many, and both, all?

Leaving all behind

- Upon further reflection, some enterprising syntacticians hit upon the idea that rather than floating all to its position next to VP, all might instead have been “left behind” by a subject that had moved.
- will [all [the students]] take the exam.
- [all [the students]], will i, take the exam.
- [the students], will [all i] take the exam.
- And why would all the students have been down there? Well, that would simplify assignment of θ-roles.
The VP-Internal Subject Hypothesis

- The verb (head of VP) can assign θ-roles to other things within the VP, which is a natural explanation for how the choice of verb controls whether an Agent θ-role is assigned or not.
- This idea became known as the **VP-Internal Subject Hypothesis**.

Back to the present

- The basic components of the quantifier “stranding” phenomenon are:
  - All the students is a constituent. The students is an NP inside all the students.
  - [all [NP the students]]
  - Either all the students or just the students can move to SpecTP to satisfy the [uN*] feature of T.
  - So all the students and the students are both NPs.
  - [NP all [NP the students]]
  - So all is essentially a noun, but one that takes an NP complement (all: [N, uN*, …]).
  - We're assuming here that all is not an adjunct, but in fact a head, taking the NP as a complement. Why?

All the students will take...

- We start by building our vP.
  - Merge the NP the exam and the V take (checks [uN*] on V)
  - Merge v and VP (HoP)
  - Move V to v (checks [uV*] on v)
  - Merge the N all and the NP the students (checks [uN*] on all)

All the students will take...

- We Merge the M will with vP (HoP)
  - This values [uInfl] on v as [uInflM].
All the students will take...

- We Merge the T with MP (HoP)
  - This values [uNfl:] on M as [uNfl:pres*] (strong).

```
T'   MP
  |     |
M   < V > NP
  |     |
N   VP
```

All the students will take...

- We move M up to T
  - This checks the strong [uNfl:pres*] on M.

```
T'   MP
  |     |
M   < V > NP
  |     |
N   VP
```

All the students will take...

- Now, there are two possibilities:
  a) Move the NP all the students.
  b) Move the NP the students.

```
M+T   MP
  |     |
M   < V > NP
  |     |
N   VP
```

The students will all take...

b) Move the NP the students.

```
TP
  |
NP
```

Expletive constructions

- An **expletive** is an element that can be in subject position without having received a θ-role from anywhere.
- **It** had been raining.
- **There** were fans rioting on Comm Ave.
Expletive constructions
1) There were fans rioting on Comm Ave.
2) Fans were rioting on Comm Ave.

The Big Picture
• Now that we’ve gotten some idea of how the system works, let’s back up a bit to remind ourselves a bit about why we’re doing what we’re doing.
• People have (unconscious) knowledge of the grammar of their native language (at least). They can judge whether sentences are good examples of the language or not.
• Two questions:
  • What is it that we know?
  • How is it that we came to know what we know?

History
• In trying to model what we know (since it isn’t conscious knowledge) some of the first attempts looked like the phrase structure rules above (Chomsky 1957).
  • 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.

Affix Hopping
• 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 the structure to get it right.
• Empirically, tense, perfect, and progressive be each control the form of the verbal element to their right.

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

Affix Hopping
• Affix Hopping
  SD: afx verb
  SC: verb+afx
  The affixes all “hop to the right” and attach to the following word.
  An ancestor to the kinds of movement rules and of course the Agree operation we’ve been talking about.
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.
- Chomsky (1970) proposed that there actually is only a limited set of phrase structure rule types.
- For any categories X, Y, Z, \( \forall v \), there are only rules like:
  - \( XP \rightarrow YP X' \)
  - \( X' \rightarrow X' WP \)
  - \( X' \rightarrow X ZP \)
- 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.

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”.
  - \( XP \rightarrow YP X' \)
  - \( X' \rightarrow X' WP \)
  - \( X' \rightarrow X ZP \)
- 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.

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.

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.
- What the system is designed to do is assemble grammatical structures where possible, given a set of lexical items to start with.
- 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:
  \[ f = G \frac{m_1 m_2}{r^2} \]
- 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.