Go back and think about the DS of *Bill saw Mary*. Where is it?

DS isn’t an interface level. We don’t want to have a DS with special properties. And we don’t any more.

Even clearer if you continue... *Sue said that Bill saw Mary.*

There is *no more D-structure* because by the time we get to the “base positions” of elements in the higher part of the tree, stuff in the lower part of the tree has already moved around.

Therefore: There can’t be any constraints that hold of DS—DS doesn’t exist (and isn’t an interface level). What was DS good for before? Can we do that in another way?

**Order by c-command:** *Linear Correspondence Axiom*

**Merge** of [] and [] results in *(wlog) {} , {}, {}*. Says nothing about *order*, only *hierarchy*. But PF needs *order*. Things need to come out in *sequence*.

And it doesn’t seem like it’s simply which projects;

```
  Drink projects in each case here, but it precedes its sister first, then follows its sister second.
```

![Diagram]

If we look at what c-commands what, we notice that where [] asymmetrically c-commands [] ([] c-commands [], [] doesn’t c-command []) [] *precedes* [].

- *I* precedes *drink* (head)
- *I* precedes *coffee*.

**Linear Correspondence Axiom** (Uriagereka’s version):

A category [] precedes a category [] iff

(a) [] asymmetrically commands [];
(b) [] precedes [] and [] dominates [].
Actually, we’re not quite done yet—see why this won’t work?

Uriagereka discusses *I like it* in this connection in a part that wasn’t assigned, suggests that Morphology linearizes on the way to PF and *wordifies* *like’t*.

**Doesn’t help with *I drink coffee*—what’s going on?**

Fine, then how about *I drink the coffee*? Ah, interesting.

Hitting the bottom of the tree—bottom Merge can never be linearized. Hmm.

**Historical note:** The LCA is actually proposed in Kayne (1994) but not in a bare phrase structure system; he uses it to derive (in a different way) many properties of X-bar theory and start an interesting research program of his own with his underlying SVO story…

Isn’t the LCA patently false?

The subject always asymmetrically c-commands the object, and assuming a DP object, this basically predicts that *all languages are SVO*.

But not even *most* languages are SVO

( most languages are SOV, lots are SVO, some are VSO…)

So what do we do with VSO?

Well, our options have been narrowed—we’re forced to the view that the verb has moved over the subject, that it asymmetrically c-commands it on the surface, even though the relation was the reverse initially.

(1) Leanann an t-ainmní an briathar i nGaeilge
Follow.pres the subject the verb in Irish
“The subject follows the verb in Irish”
This gets even more dramatic for SOV languages—
(and they’re usually actually SOVTC languages!)

Points (from Uriagereka):
• Anaphors in English are clausemate bound
• Anaphors in Japanese seem not to be—they seem to be able to relate to subjects which are further away than in English.
• But embedded clauses are actually structurally kind of close to the subject in Japanese.

Economy:
• The need to express a meaning never drives movement—
even if in English we wanted the meaning with the anaphor taking a more distant antecedent, we aren’t allowed to exceptionally do a Japanese-like movement. It’s not forced in English, it’s not possible in English.

Covert movement

If • Irish VSO is derived from SVO via verb movement
and • All languages have equivalent-looking LF’s
the • The verb has to move in English before LF.

We don’t see it, so in English it must be after Spell-out.
Feature movement

Movement is *driven* by the **features** of the lexical items we are combining.

One question that comes up: Why do whole lexical items move? Why do whole *phrases* move? If CHL only *cares* about features, why doesn’t it move the features around alone?

It’s possible that we might think you can’t move a feature out of its bundle, that you can only find a bundle with the right feature and move it—that might answer the first question.

A perhaps related question is: Why Procrastinate? Why is it “cheaper” to do movement after Spell-out?

Chomsky’s idea about this was that the only reason we move whole phrases around is really a fact about the PF interface—you can’t pronounce features by themselves. If people could communicate with telepathy, this might be unnecessary, but because we need to pronounce things, we need all of the features to be together (not scattered around the tree).

However, *after* Spell-out, this is no longer a concern. Features are free to move by themselves.

Uriagereka puts it thusly: lexical items are glued together prior to Spell-out, but Spell-out rips the lexical item apart anyway (sending its phonological features off to PF), so maybe at that point the formal features and semantic features may also be unglued. And the formal features can then move around on their own.

Another example (both intended as an explanation of why lexical items move and in fact why sometimes even phrases move): *Whose book is this?*

We move *Whose book*, even though presumably it’s just the *who* that *needs* to move.

(3)  

```
    DP
   /   |
DP   D□
  /  |
D  NP
  |
′s  book
```

We could move *who* by itself, but then we’d be left with *′s book*, which can’t be pronounced (*who* and *′s* form a unit as far as PF is concerned). So, we move the smallest thing (a complete object, a constituent—so not just *whose* either) that we can.
An influential stage in the MP made use of this idea of **feature movement** as being the basic form of movement, and moving more than just a feature to be forced by phonological concerns. Movement was driven by the operation **Attract-F** (find a feature to eliminate the uninterpretable feature you are concerned with, and move it up to you) but had to sometimes be supplemented with a full **Move** because otherwise the result would be uninterpretable at the PF interface.

**Uriagereka, Watanabe, Japanese:**
- Japanese has a null *wh* which stands and moves alone, overtly.
- Perhaps..

**Move**

Merge as a binary transformation; Move as a singulary transformation (possibly able to target subparts of phrase smaller than lexical items).

Necessarily extends the structure;
- Take an object K, pick a subpart P, Merge K and P.
- K will project; generally the thing that needed something projects.

Necessarily c-commands...

**Attract F**

In fact, things become even a little bit simpler if we think of movement as “attraction” instead. That is, when a feature moves, it moves from lower in the tree to higher in the tree. Consider the EPP again. It says that “SpecTP must be filled (with a nominal element)”. This is a property of T—it has some uninterpretable feature that must be checked off. That uninterpretable feature didn’t even get into the derivation until we introduced T.

Moreover, if that feature is [strong], then we need to take care of it right away.

Under an “Attract” view, what happens is this: We introduce T into the tree and then look down into the tree for something that would be able to eliminate the uninterpretable feature. You only look until you find something, and once you find something, you move that. This gets **shortest move** effects, again following from a kind of least effort, but it gets it more directly. *Wh*-movement in English just looks down into the IP until it sees its first [wh] feature, and then moves that feature to C/SpecCP: Superiority.
The differences are a little bit subtle between a Move F and an Attract F approach, but Attract F feels more intuitive and is often the way minimalist discussions are cast.

---

**Head-movement and the lexicalist hypothesis**

Another thing that differs somewhat between the MP and previous ways of doing things. First of all—there’s no downward movement—there’s no way to do it. There must not be any kind of Affix Hopping. The old way was:

\[(4)\]

\[
\begin{array}{c}
  T \quad VP \\
  [past] \quad [past] \\
  -ait \quad -ed \\
  \quad V \\
  \quad V
\end{array}
\]

\[(French)\]

\[(English)\]

This can’t be the way it works anymore. The hypothesis (“the lexicalist hypothesis”) in the 1993 MPLT paper was that verbs come out of the lexicon **pre-inflected**. A past tense verb comes into the numeration with its past-tense pronunciation already settled.

In English is where it matters—the verb surfaces with the tense and agreement morphology (\textit{says}, \textit{said}), and hence (uninterpretable) tense and agreement features. They must be checked (the origin of the term “check”—we don’t want a [past] T and a [present] V), but this happens after Spell-out.

This is assumed to be the way it works in French too for uniformity—but it isn’t required.

This is a move away from the **one morpheme one head** approach that characterized earlier stages of the generative approach.

(A more recent take on this is Distributed Morphology…)

---

**There is no AgrP**

There are four main kinds of functional category (i.e. non-lexical category—\textit{N}, \textit{V}, \textit{Adj}, and maybe \textit{P} are lexical) that we’ve been dealing with: \textit{D}, \textit{C}, \textit{T}, and Agr. But the first three have interpretive consequences. Agr seems only to be there in order to give us word order and agreement—it doesn’t appear to have an interpretable features.
The way Agr works seems to be that a verb, with its uninterpretable [-] features (agreement) adjoins to the head, and then an argument, with its interpretable [-] features moves to SpecAgrP and checks off the uninterpretable [-] features on the verb. That is, Agr is only there to mediate feature checking.

Our only evidence for Agr seems to be from the cases where its features are strong—when there is overt movement to SpecAgrP. So, perhaps AgrP is only there when its features are strong. Can we say that AgrP is there just when it is strong?

Consider object shift, starting here:

(5)
```
  \node (vmax) at (0,0) {$v^\text{max}$};
  \node (subj) at (-2,1) {\text{Subj}};
  \node (v) at (-3,0) {$v{\hbox{-}}\text{[]}\ $};
  \node (v'p) at (-2,-1) {\text{VP}};
  \node (v) at (-1,-1) {$V\ $};
  \node (obj) at (1,-1) {\text{Obj}};
```

Shifting the object into SpecAgr would result in something like this (and we have overt evidence of this position in many cases): adding an Agr with a strong [D] feature to move the Object and a strong [V] feature to move the verb.

(6)
```
  \node (agrp) at (0,0) {AgrP};
  \node (obj) at (-2,1) {\text{Obj}};
  \node (agri) at (-3,0) {Agr\{\hbox{-}}\ $};
  \node (agrv) at (-1,0) {Agr\ +\ v\ $};
  \node (vmax) at (0,0) {$v^\text{max}$};
  \node (subj) at (-2,1) {\text{Subj}};
  \node (v) at (-3,0) {$v{\hbox{-}}\text{[]}\ $};
  \node (v'p) at (-2,-1) {\text{VP}};
  \node (v) at (-1,-1) {$V\ $};
  \node (t_v) at (1,-1) {t_v};
  \node (t_obj) at (1,-2) {t_{\text{Obj}}};
```

But couldn’t we as easily just put the strong [D] feature on $v$? Even the subject has a [D] feature, it isn’t in the checking domain of $v$ for the simple reason that it was Merged there—the foot of even a trivial chain is not in the checking domain of the head it is merged with (this was discussed elsewhere in the chapter). So, instead, we have this, where the object just moves into a second specifier of $v^\text{max}$ (still able to check the strong feature from there).
Assuming that adverbs can come between the shifted object and the subject, we can still maintain this structure, assuming that adverbs also occupy SpecP and that the only requirement on strong features is that they be checked before leaving the projection (i.e., before merging vP with something else).

In the rare cases that really motivate AgrSP (multiple subject constructions, like in Icelandic, where you can have an expletive and a subject at the beginning of the sentence), we can do the same thing, assume that T has two specifiers in that case.

**Merge before Move**

The EPP can be satisfied either by moving the subject into SpecTP or by inserting an expletive there.

(8) There is a fly in the soup.
(9) A fly is in the soup.

We used to think of the there cases as insertion of an expletive, but we can’t anymore—we can’t add anything that wasn’t in the numeration. So, these two differ with respect to whether there is in the numeration.

In both cases we get to a stage like this, more or less:

(10)
and the next step is determined by whether our numeration still contains *there* or not. If there is a *there*, it gets Merged (which is assumed to check the [D] feature of T), and if not, a *fly* is copied and Merged.

If we have *there* and move a *fly* anyway, the derivation crashes—we don’t have a single object left at the end for LF to interpret, we have two (the tree, and *there*).

(11) There seems to be a fly in the soup.
(12) A fly seems to be in the soup.

Consider (12) first. Previous knowledge tells us that it works like this:

(13) [A fly] seems [a fly] to be [a fly] in the soup.

There are two movements, one to the lower SpecTP, then another to the highest.

Suppose we have *there* in the numeration. At the first step, in principle, we could insert *there* or move a *fly*.

If we insert *there*, we get (11). On the second step, no *there* remains in the numeration, the closest thing that could satisfy the EPP is moved, so *there* moves to the top.

(14) There seems there to be [a fly] in the soup.

What if we’d taken the other option? Moved a *fly*, thus leaving *there* in the numeration (and then forcing it to be inserted in the higher SpecTP?)

(15) * There seems [a fly] to be [a fly] in the soup.

The result is ungrammatical. That wasn’t an option, it seems.

Following the least effort intuition, we suppose that (15) is bad because (14) is “cheaper”. At the first step, when we had the option either to Merge or to Move, we had to choose Merge.
Why might Merge be cheaper?

Well, Move actually is Merge and some other stuff, like Find and Copy. So, the stuff Move has to do is a proper superset of the things Merge has to do. Given that, Merge should be preferred if the system is maximizing economy.

So Merge always takes precedence over Move.

What does that mean, Merge before Move?

Idea: Syntax is driven by various pressures of economy. One of them is “do easier things before harder things”

- Procrastinate (do all movement after Spell-out where possible)
- If you can either Merge or Move, Merge (since Move includes Merge).

So there is a comparison going on here—between what?