The basic proposals:

X-bar structures (universally) have a strict order: Spec-head-complement.
There is no distinction between adjuncts and specifiers.
There can be only one specifier.

But wait!—What about SOV languages? What about multiple adjunction?
Answer: We’ve been analyzing these things wrong.
Now, we have lots of work to do, because lots of previous analyses relied on
the availability of “head-final” structures, or multiple adjunction.

Why make our lives so difficult? Wasn’t our old system good enough?
Actually, no.
A number of things had to be stipulated in X-bar theory (which we will review);
they can all be made to follow from one general principle.
The availability of a head-parameter actually fails to predict the kinds of languages
that actually exist.
A more restrictive grammar is simpler—perhaps slightly closer to plausible in
terms of acquisition.

Up through the early 1990’s, people generally considered syntactic structures to be kind of
like “mobiles”—c-command, dominance, hierarchy was important, but linear order was
not important. Hence, Japanese SOV sentences were in all syntactically relevant respects
just like English SVO sentences.

Kayne’s proposal says that (asymmetric) c-command and linear order are directly
related—something that comes first asymmetrically c-commands something that comes
later. No parameter to set.
What does that mean? **Linear precedence** is the relation at issue, terminal nodes are the words, and so the LCA says that, as determined by asymmetric c-command in the tree, every word either precedes or follows every other word, but not both.

It seemed complicated, but the intuition is clear:
- Language requires articulating words in order.
  (not even forced by the language faculty—forced by the real world)
- Language ensures that this is possible.
- The formal version “it is possible to pronounce the words in order” is just as given above.

The cool thing about the LCA is that a number of things we used to have to stipulate about X-bar theory now just come for free. It couldn’t have been any other way.

**Labeling:**

- A **head** is a nonterminal that dominates no other nonterminals.
- A **nonhead** is a nonterminal that dominates at least one other nonterminal

There is an immediate problem: (7) is ruled out.

And this is a big problem, since (7) is the only X-bar structure there’s supposed to be.

(7)  
\[ \ast \]  

\[
\begin{array}{cccc}
YP & Y' & X' & ZP \\
Y & X & Z' \\
y & x & z
\end{array}
\]

\[ A = \{ <YP,ZP>, <YP,Z'>, <YP,X'>, <X',Y'>, <X,Z'>, <X,Z> \} \]

so \( d(A) = d\{ <YP,ZP>, <YP,X>, <X',Y'>, <X,Z'> \} \)

The problem lies in the fact that we have both \( <X',Y'> \) and \( <YP,X> \) in there.

(8)  
\[ \ast \]  

\[
\begin{array}{cccc}
YP & X' \\
ZP \\
y & x & z
\end{array}
\]

\[ d(A) = d\{ <YP,ZP>, <YP,X>, <X',Y'>, <X,Z'> \} \]

\[ d<YP,ZP> = \{ <y,z> \} \quad \text{"y precedes z"} \]

\[ d<YP,X> = \{ <y,x> \} \quad \text{"y precedes x"} \]

\[ d<X',Y'> = \{ <x,y> \} \quad \text{"x precedes y"} \]

\[ d<X,Z'> = \{ <x,z> \} \quad \text{"x precedes z"} \]

so \( d(A) = \{ <y,z>, <y,x>, <x,y>, <x,z> \} \)

\[ d(A) = \{ <y,z>, <y,x>, <x,y>, <x,z> \} \]

\[ d(A) = \{ <y,z>, <y,x>, <x,y>, <x,z> \} \]

\[ z \text{ follows both } x \text{ and } y. \]

\[ y \text{ precedes both } x \text{ and } z. \]

\[ \text{but } x \text{ also precedes } y...! \]

This won’t do; we haven’t even gotten off the ground.

Obviously, the one we need to get rid of is \( <X',Y'> \).

We need it to turn out that \( X' \) does not asymmetrically c-command \( Y' \).

**The solution.** Suppose that the X-bar structure arises from adjoining \( YP \) to \( X' \) (which we might as well call “XP” now), and define c-command such that XP won’t c-command YP.

(9)  
\[ \alpha \text{ c-commands } \beta \text{ if and only if:} \]

- \( \alpha \) and \( \beta \) are categories (that is, not a segment unless it is the top segment)
- no segment of \( \alpha \) dominates \( \beta \) (“\( \alpha \) excludes \( \beta \)”), and
- every category that dominates \( \alpha \) also dominates \( \beta \).

In English: **adjuncts c-command what they adjoin to, but not vice-versa**

(below, XP c-commands YP, but XP doesn’t c-command YP).

(10)  
\[ \]  

\[
\begin{array}{cccc}
YP & XP \\
Y' & X' & ZP \\
Y & Z' \\
y & x & z
\end{array}
\]

\[ d(A) = d\{ <YP,ZP>, <YP,X>, <X',Y'>, <X,Z'> \} \]

\[ d<YP,ZP> = \{ <y,z> \} \quad \text{"y precedes z"} \]

\[ d<YP,X> = \{ <y,x> \} \quad \text{"y precedes x"} \]

\[ d<X',Y'> = \{ <x,y> \} \quad \text{"x precedes y"} \]

\[ d<X,Z'> = \{ <x,z> \} \quad \text{"x precedes z"} \]

so \( d(A) = \{ <y,z>, <y,x>, <x,y>, <x,z> \} \)

\[ d(A) = \{ <y,z>, <y,x>, <x,y>, <x,z> \} \]

\[ z \text{ follows both } x \text{ and } y. \]

\[ y \text{ precedes both } x \text{ and } z. \]

\[ d(A) \text{ is a linear order.} \]
Now, we can “derive X-bar theory”…

- Every phrase must have at least one head.
- No phrase can have two heads.
- The complement of a head cannot be a head (must be a phrase)
- A phrase can have only one specifier.
- A phrase cannot be adjoined to a head.
- A head cannot be adjoined to a phrase.
- A specifier cannot be a head (it must be a phrase).
- “X′” cannot be moved by itself, leaving the specifier behind.
- A head to another head will c-command the c-command domain of the original head (a moved head will c-command its trace).

With some extra things as well…

- There are no specifiers at all—there are only adjuncts; there is no distinction between an adjunct and a specifier.
- Multiple clitic adjunctions cannot be analyzed as multiple adjunction to, say, T—they must be adjoined to one another, then adjoined to T.
- "Multiple specifiers” can exist in a similar way, adjoined to one another, allowing an explanation for the apparent ability for the highest specifier to “c-command out” (more below).

We won’t go through all of the cases; they’re worked out in the book, and the logic is always the same. Property X (say, “A phrase can only have one head”) follows because if X were violated, the resulting tree would be “too symmetric” (e.g., the heads would c-command each other and thus neither would precede the other).

Some highlights…

Two-headed phrase a phrase with a head for a complement

\[ d(A) = d(\langle B,Z \rangle, \langle B,X \rangle) \]

\[ d\langle B,Z \rangle = \{ \langle b,z \rangle \} \]  “b precedes z”

\[ d\langle B,X \rangle = \{ \langle b,x \rangle \} \]  “b precedes x”

so \[ d(A) = \{ \langle b,z \rangle, \langle b,x \rangle \} \]

\[ b \text{ precedes both } x \text{ and } z. \]
\[ \text{ but } x \text{ neither precedes } z \text{ nor vice-versa.} \]
\[ d(A) \text{ is not a linear order.} \]

Adjoining a head to a head (e.g., head movement)

(12)

\[ T \quad \text{TP} \quad \text{YP} \]

\[ X \quad T \quad Y \quad Z \quad \text{ZP} \]

\[ x \quad t \quad y \quad z \]

Notice: X c-commands YP—
and X asymmetrically c-commands Y, ZP.

- X and YP are categories (not segments).
- No segment of YP dominates X.
- Every category that dominates X also dominates YP.
- —T? No, the category T does not dominate X, only one segment does. —TP. Yep, and it dominates YP.

So, if X was a head moved from lower in the tree (say, somewhere in ZP), X would c-command its trace—as is proper.

There are no specifiers, there are only adjuncts—but you can’t adjoin more than one phrase to a phrase. Here’s why:

Adjoining two things to the something

(13)

\[ T \quad \text{XP} \quad \text{YP} \quad \text{XP} \]

\[ y \quad z \quad x \]

The problem here is that:

B c-commands A

- B and A are categories
- No segment of A dominates B.
- Every category that dominates B dominates A.

and

A c-commands B

- A and B are categories
- No segment of B dominates A.
- Every category that dominates A dominates B.

so

Neither asymmetrically c-commands the other.

Same problem between YP and ZP.

A problem: Plenty of examples of multiple clitics, usually assumed to head-adjoin to a head high in the tree (T, AgrS).

(14) Jean vous le donnera.

J. youDAT it will-give

‘Jean will give it to you.’
This forces us to a different analysis—either they adjoin to different heads or one clitic adjoins to another. Above, vous adjoins to le and then they both adjoin to AgrS, for example.

\[(15)\] AgrSP * AgrSP

\[\text{AgrS TP AgrS TP} \]

\[\text{le} \quad \text{AgrS} \quad \text{...} \quad \text{le} \quad \text{AgrS} \quad \text{...} \quad \text{vous} \quad \text{AgrS} \]

An added bonus: Specifiers seem to be able to c-command out of their phrase—below, every girl can bind the pronoun she, which requires c-command. Supposing the specifiers are adjuncts and our definition of c-command allows it to work:

\[(16)\] C

\[\text{DP}_1 \quad \text{TP} \quad \text{TP} \quad \text{DP}_2 \quad \text{DP}_1 \quad \text{T} \quad \text{VP} \quad \text{every girl} \quad \text{'s} \quad \text{D} \quad \text{thinks she is a genius} \quad \text{father} \]

**Implications for word order**

We decided that asymmetric c-command means precedes. Why couldn’t it mean follows? Could it be a parameter?

Well, one thing we know: *Precedes or follows*, the specifier is on the opposite side of the head from the complement.

So, if it’s a parameter, that would allow SVO and OVS orders. It would not provide an explanation for the world’s most common order, SOV.

So, since a parameter isn’t going to help us, and since SVO is very, very much more common than OVS as a surface order, precedes is almost certainly the right choice. Plus, SpecIP and SpecCP are nearly invariably initial.

**SOV order**

Let’s think about Japanese for a moment…

\[(17)\] John-ga [Mary-ga hon-o kat-ta to] omotteiru

\[\text{Japanese} \quad \text{John- NOM Mary- NOM book- ACC bought-past that thinks} \]

\[\text{T} \quad \text{C} \]

Japanese is a uniformly head-final. Verbs are last, T is last, C is last. But of course, we aren’t allowed to “heads are final” despite appearances. What this means... something like this has to be going on:

\[(18)\] a. VP b. TP

\[\text{Obj} \quad \text{VP} \quad \text{VP} \quad \text{i} \quad \text{T} \quad \text{j} \quad \text{C} \]

\[\text{Obj} \quad \text{V} \quad \text{i} \quad \text{T} \quad \text{j} \quad \text{C} \]

\[\text{Crazy…? Isn’t that a lot of movement for the most common type of language?} \]

**Consider:** Languages that move a wh-word (to the left) seem to be head-initial. Languages that are not head-initial seem not to move their wh-words.

(SOV languages seem to be overwhelmingly wh-in-situ languages)


Why might that be?

Well, wh-movement takes wh-words into SpecCP, right?

**But in SOV languages, SpecCP is full—it has IP in it.**
Koopman and the Spec-head configuration

Koopman’s paper goes basically like this: What kinds of possibilities does Kayne’s proposal suggest to us? If we adopt certain additional restrictions on our theoretical machinery, what kinds of things will this force us to do with respect to existing analyses?

Koopman reviews several potential cases of “heavy pied-piping”—movement of very large phrases leftward (like, for example, inverting the entire IP with C as above).

One example—perhaps the most convincing—comes from Nweh, which appears to be a very regular, “head-initial” language—except for the negative morphemes.

(19) a. Njikem a ke? te pfet akendɔŋ bɔ. N. Agr P-1 neg eat plantains neg
    ‘N didn’t eat plantains.’

    b. Njikem a ke? te akendɔŋ pfet. N. Agr P-1 neg plantains eat
    ‘N didn’t eat plantains.’

The fact that the verb can substitute for bɔ here strongly suggests that it is a head—this means that if we are going to make a parallel to the well-known case of bipartite negation (ne…pas in French), we must equate bɔ with ne and te with pas.

How does the VP get to the left of the neg head in (19a)? Nweh is everywhere else uniformly head-initial, so Kayne or not, we’re led to suppose that it moves there below negation.

(20) a ke? [AP [VP te pfet akendɔŋ]] AP [NegP bɔ] rVP ...

And this moving VP can be huge—even carrying along a whole embedded clause.

(21) n ke? [te ju? le Njikem a ka fia nkap ambo Atem ajua]₁ bɔ ɪ₁ I P1 TE hear that N Agr P2 give money to Atem yesterday bɔ
    ‘I did not hear that N gave money to Atem yesterday.’

Another possible example is intonation-only questions in French, where we have a small amount of evidence that the IP has moved into SpecCP.

First, (some head in) C is the home of the [+Question] feature that distinguishes questions from statements. This can be realized just with intonation, but if it is, it is realized at the end of the question—something like the Nweh bɔ case.

(22) [CP [IP tu venir demain]]₇ Q[INTON] ɪ₇ ‘Are you going to come tomorrow?’

Koopman notes one further piece of evidence for this (from Dominique Sportiche):

Like in English, “negative polarity items” (NPI) like anyone, ever, at all are licensed in questions. The way NPI’s are generally supposed to be licensed is by c-command; an NPI is ok in the c-command domain of a licensor.

(23) a. I didn’t see anyone.
    b. I didn’t sleep at all.
    c. I didn’t say that Mary told me she bought anything.

(24) a. DidQ you see anyone?
    b. DidQ you sleep at all?
    c. DidQ you say that Mary told you she bought anything?
    d. I asked John ifQ Mary bought anything.

    b. * I slept at all.
    c. * I said that Mary told me she bought anything.

(26) a. * Anyone didn’t see me.
    b. * I told anyone that Mary didn’t buy coffee.
    c. * I asked anyone ifQ Mary bought coffee.

In French intonation-only questions, NPI’s are not licensed—suggesting that they are above the would-be licensor Q:

(27) a. AQ-t-il vu qui que ce soit?
    has-t-he seen anyone
    ‘Has he seen anyone?’

    b. * [IP 1a vu qui que ce soit]₁ [INTON]Q ɪ₁ he has seen anyone
    (‘Has he seen anyone?’)
Koopman proposes modifying Kayne’s system in a particular way:

Kayne was troubled by the fact that the canonical X-bar structure was ruled out.

\[
(7) \quad * \quad XP
\]

\[
YP \quad X' \quad ZP
\]

\[
Y' \quad X \quad Z' \quad Y \quad Z \quad Y
\]

…and he responded by saying that specifiers are adjuncts, and the c-command relation was such that adjuncts c-command the node they are adjoined to, but not vice-versa.

However—the intuition behind the LCA is that things need to be pronounceable—Any words you pronounce need to either precede or follow the others…

It would then follow that there is no particular reason that an unpronounced element needs to be assigned a place in the linear order.

**Koopman’s solution:** (7) is only ruled out if both \(x\) and \(y\) are pronounced.

**Hence:** No phrase can have both a pronounced head and a pronounced specifier.

In fact, this sounds a lot like the Doubly-filled COMP filter that prohibits whether that.

—but now the proposal is that it is very general, true of all pronounced structures.

An immediate problem: There seem to be cases where both Spec and Head are pronounced. (These must have been mis-analyzed, we need a new analysis).

(28) Who did you see? Isn’t that just did in C and who in SpecCP?
(29) Morgen komt Jon ‘Tomorrow, Jon comes’ (Dutch, V2)

Consider V2: People usually take this to be movement of the verb into some C position, and then movement of some XP into the specifier of that C position (Top°?)

Under Koopman’s plan, XP must be in a higher projection than the verb.

We are forced to assume that CP consists of multiple projections—but we already believe that this is necessary, according to Rizzi (1995).

Embedded wh-questions in Dutch…

(30) Ik vraag me af wie (of) (dat) er morgen komt
    I wonder who if that there tomorrow comes
    ‘I wonder who is coming tomorrow.’

(31) [ wie WH° [ — Q° [ — C° [ …
    of dat

and assuming that structures are basically invariant across languages, we can describe English too…

(32) [ who WH° [ — Q° [ — C° [ …
    did

So, there are a number of existing cases that need to be rethought, the general logic being: Where we used to think we had both an overt specifier and an overt head, what we really have is two phrases…

(33) Not XP but rather XP

\[
\text{over} \quad X' \quad \text{over} \quad X' \quad YP
\]

\[
\text{over} \quad X \quad \text{over} \quad Y
\]

Koopman points out a number of cases where thinking of things this way might be fruitful, although all of the suggestions are fairly speculative. Again, some highlights:

VSO languages:

In VSO languages, complementizers and negation appear initially, before the verb. Head-movement in Kayne’s system requires that the moving head adjoin to the left of the hosting head—hence, we know that the verb has not moved as high as negation and the complementizers.
Another (fairly well-known) property of Irish (the canonical VSO language) is that when there is no subject, the verb shows agreement—whereas when there is a subject, the verb does not show agreement.

Koopman proposes that:

- In Irish, the verb moves all the way up to AgrS.
- AgrS is where subject agreement is effected (by Spec-head agreement).
- If the subject is overt, it can’t land in SpecAgrSP because then there would be both an overt head (with the verb in it) and an overt specifier. The subject stops short of SpecAgrSP, perhaps staying in SpecTP.
- If the subject is null (pro), it can get up to SpecAgrSP and trigger agreement.

“Disappearing heads” in Italian. It appears that non can appear except when a negative constituent is to the left of the finite verb.

(34)  a. Nessuno è venuto
Nobody has come

   b. * Nessuno non ha visto Mario.
      nobody not has seen Mario
      (‘Nobody has not seen Mario.’)

   c. Non ho visto nessuno.
      Not have seen nobody.
      ‘I have not seen anybody.’

   d. Niente ho detto.
      Nothing have said
      ‘Nothing, I have said.’

If we look at this as a situation for the Neg-Criterion, nessuno has moved into SpecNegP, non is in Neg°, and they can’t both be overt.

Exploring a further constraint:

(35)  A projection is interpretable if and only if it is activated by lexical material.

This means that some lexical item needs to pass through any functional projection that is going to participate in interpretation—this gives “motivation” to the large amount of movement that seems to occur in languages under the LCA approach.

Some languages show evidence that root clauses are CP (markers of declaratives; e.g., Korean), so if we assume that this is therefore true of all languages, and given that English does not show an overt complementizer, English main clauses must look like:

(36)    [CP [IP … … ] i C ]
(no evidence given, but it’s forced)

That-trace effects and subject wh-questions…

(37)  a. Who came yesterday?

   b. * Who did come yesterday?

(38)  a. Who did you think came yesterday?

   b. * Who did you think that came yesterday?

We know that subject extraction is crosslinguistically difficult. Let’s suppose that (39) correctly describes this situation (whatever it might follow from):

(39)  A wh-phrase can never be extracted from the Spec position immediately c-commanded by C.

So, in a subject wh-question, the wh-word can’t move to SpecCP, but something needs to. It must be the IP, then.

(40)    [CP [IP ] i C ]
   ? *No do-support based on proper binding condition…??

And for (38), the same happens in the embedded clause, and then who moves out from there. There can be no that because that would mean both C and SpecCP are overt.

(41)    [CP [IP ] i C ]
   ?

Incidentally, this makes some sense of questions like:

(42)  Who came do you think?

(43)  [ [CP who came ] i C ]
   ??